NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UAV SWARM BEHAVIOR MODELING FOR EARLY EXPOSURE OF FAILURE MODES

by

Michael B. Revill

September 2016

Thesis Advisor: Kristin Giammarco
Co-Advisor: Mikhail Auguston

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2016	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE UAV SWARM BEHAVIOR MODELING FOR EARLY EXPOSURE OF FAILURE MODES			**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael B. Revill			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Over the past decade, the Department of Defense has placed a great amount of attention on the advancements of unmanned aerial vehicles (UAVs), and more specifically on employing a large number of autonomous UAVs into "swarms." These swarms form an organized cluster of vehicles to act out multifaceted operations as a group. Despite the benefits offered by UAV swarms, there are hurdles that engineering teams must grapple with while designing a UAV swarm system. One key area is creating and understanding the swarming behavior and revealing all potential failure scenarios that may impact the desired mission. This research uses Monterey Phoenix (MP) to model system behaviors by grouping them into distinct, reusable agent-like models of possible actor behaviors and modeling actor interactions as separate constraints. This approach affords the ability to compute every possible variation of actor behaviors with every other possible actor behavior from these models, which generates an exhaustive set of possible scenarios or event traces. Through manual inspection or semi-automated assertion checking of these event traces, the discovery of unwanted and undesirable behaviors and failure modes is achievable, which allows mission planners to then counteract these unsolicited instances with necessary failsafe behaviors.

14. SUBJECT TERMS swarm, search and rescue, behavior modeling, UAV, Monterey Phoenix, failure modes, failsafe behaviors			**15. NUMBER OF PAGES** 109
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

UAV SWARM BEHAVIOR MODELING FOR EARLY EXPOSURE OF FAILURE MODES

Michael B. Revill
Civilian, Department of the Navy
B.S., Limestone College, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2016

Approved by: Kristin Giammarco, Ph.D.
Thesis Advisor

Mikhail Auguston, Ph.D.
Co-Advisor

Ronald Giachetti, Ph.D.
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Over the past decade, the Department of Defense has placed a great amount of attention on the advancements of unmanned aerial vehicles (UAVs), and more specifically on employing a large number of autonomous UAVs into “swarms.” These swarms form an organized cluster of vehicles to act out multifaceted operations as a group. Despite the benefits offered by UAV swarms, there are hurdles that engineering teams must grapple with while designing a UAV swarm system. One key area is creating and understanding the swarming behavior and revealing all potential failure scenarios that may impact the desired mission. This research uses Monterey Phoenix (MP) to model system behaviors by grouping them into distinct, reusable agent-like models of possible actor behaviors and modeling actor interactions as separate constraints. This approach affords the ability to compute every possible variation of actor behaviors with every other possible actor behavior from these models, which generates an exhaustive set of possible scenarios or event traces. Through manual inspection or semi-automated assertion checking of these event traces, the discovery of unwanted and undesirable behaviors and failure modes is achievable, which allows mission planners to then counteract these unsolicited instances with necessary failsafe behaviors.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CONOPS	concept of operations
DOD	Department of Defense
DODAF	Department of Defense architecture framework
DRM	design reference mission
FALT	falconry alternate lure training
GCS	ground control station
GPS	global positioning system
GT	greater than
LT	less than
M&S	modeling and simulation
MP	Monterey Phoenix
PID	person(s) in distress
RTL	return to launch
SA	SAR assets
SAR	search and rescue
SOP	standard operating procedures
SOS	system of systems
UAS	unmanned aerial system
UAV	unmanned aerial vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Since 2001, on an international scale, the United States military has had over 400 unmanned aerial vehicles (UAVs) and drones crash and create major accidents. The drones failed because of issues with weather, mechanical components, human error, and many other reasons (Whitlock 2014). The main goal of this research is to determine the efficacy of modeling UAV failure modes and corresponding failsafe behaviors alongside Search and Rescue (SAR) models. By placing these failure modes and failsafe behaviors into concepts of operations (CONOPS) early in the mission planning life cycle, we are able to identify pitfalls that would be otherwise costly and detrimental to the success of the mission.

Today, the Department of Defense is moving to a more distributed approach to using UAVs by employing multiple UAVs into swarms. Despite the benefits offered by UAV swarms, there are hurdles engineering teams must grapple with while designing a UAV swarm system. One key area is creating and understanding the swarming behavior and revealing all potential failure scenarios that may impact the desired mission. This research uses Monterey Phoenix (MP) to model system behaviors by grouping them into distinct, reusable agent-like models of possible actor behaviors and modeling actor interactions as separate constraints. This approach affords the ability to compute every variation of actor behaviors with every other possible actor behavior from these models, which generates an exhaustive set of possible scenarios. The amount of potential swarm behavior states produced by MP far exceeds the number that a capable human could manually generate and can be accomplished in a fraction of the time. Through manual and semi-automated inspection of these event traces, the discovery of unwanted and undesirable behaviors and failure modes is achievable, which allows mission planners to then counteract these unsolicited instances with necessary failsafe behaviors.

Some of the UAV failure modes modeled in this research include payload loss of link, loss of global positioning system (GPS), and autopilot failure. In planning for the mitigation of risks and potential failures, certain assumptions are made about what behaviors should be implemented to counteract these unintended consequences. For

example, a reasonable failsafe behavior that was initially modeled for the loss of GPS was to reduce throttle and attempt to land. After modeling this mitigation tactic and several other failure modes in MP and inspecting the various event traces, a new failure mode option became apparent. The early planning of the failsafe behavior for a payload loss of link failure mode was to attempt to re-establish connection. After modeling both loss of GPS and payload loss of link, the new failsafe behavior (attempt reconnection) was implemented into the model. This research identifies a common failsafe behavior as a recovery strategy for two UAV failure modes. After modeling all failure modes, the process of failure mode prioritization started. Given the incremental approach to modeling individual failures, it was not until this point that inconsistencies among similar failsafe behaviors started emerging.

Monterey Phoenix makes the otherwise daunting task of understanding the behavior of complex processes and systems manageable through rapid generation of event traces. Commonality of failure mode recovery strategies is just one of several benefits that MP afforded this modeling research effort. Others include the establishment of a pattern for rapid failure mode modeling, understanding the prioritization of failure modes, and a surprising emergent behavior from a bingo fuel scenario that inspired a failsafe behavior that may have been otherwise overlooked until live experimentation or testing.

References

Whitlock, Craig. 2014. When Drones Fall from the Sky. *Washington Post*. http://www.washingtonpost.com/sf/investigative/2014/06/20/when-drones-fall-from-the-sky/.

ACKNOWLEDGMENTS

I cannot express enough thanks to my thesis advisor, Dr. Kristin Giammarco, for her expertise, guidance, encouragement and patience throughout the process of developing my thesis. She never hesitated to make time available to help me with a difficult question or to get me back on track. I would like to thank Dr. Mikhail Auguston for his assistance and direction with understanding Monterey Phoenix. Similarly, I would like to thank Dr. Timothy Chung for his technical support and suggestions in the areas of swarming concepts. Without the guidance each of these professors and advisors provided, this research and resulting thesis would not have been possible.

I must give thanks to my IPT lead, Jason Livingston at SPAWAR Systems Center Atlantic, for his patience with me through this two-year process. He worked with my schedule and was very understanding of the demands that accompanied the research and development of a thesis. Additionally, I would like to thank my co-worker and friend, Paul Walter, for pushing me to pursue my master's degree.

Finally, I would like to graciously acknowledge my family, the individuals who have been directly and indirectly impacted by this extremely demanding journey. First, I am forever indebted to my wife, Erin, and my two children, Morgan and Reece. Despite what must have felt like an absentee husband and father, through the rigor and struggles of completing this thesis, they not only continuously provided support and encouragement, but also were my source of joy and inspiration. Also, my parents, Andy and Kathy, have always believed in me and were there for Erin and me when we needed support. I consider them my boosters that continuously pushed me to do my best.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM STATEMENT

The ability to understand how a single event can have cascading effects on corresponding processes within a single system or system of systems can provide invaluable benefit to a project, mission or program. This is especially true in unmanned operations, where mission planners must understand the gravity of every behavior in a large multiple agent operation. Whether in defensive, offensive or humanitarian initiatives, future and some present day operations are moving rapidly toward the employment of swarm operations to overwhelm or saturate the given objective through "deliberately structured, coordinated, strategic way to strike from all directions, by means of a sustainable pulsing of force and/or fire" (Arquilla and Ronfeldt 2013). In one-to-one unmanned systems operations (i.e., one operator and one UAV), it is relatively simple for the human brain to comprehend the multiple steps in an operation. Furthermore, it is feasible for the human brain to understand how unexpected changes or behaviors can impact a simple one-to-one ratio system. As we increase the volley of agents and start utilizing swarms to accomplish otherwise easy-to-understand tasks, on an exponential scale, the ability to calculate the impact of a single event on the swarm is lost, as demonstrated in Chapter II.

In addition to understanding the expected or anticipated events of a system, it is even more difficult to understand what impact a system component failure may have to the system. Failure of a UAV is difficult to predict. A variety of system behavior models have been used, but all have their limitations (Giammarco and Auguston 2013). Unfortunately, the mitigation planning for failure modes is too often introduced later in a project's development cycle, which can have huge impacts on cost and schedule. In swarm operations, improperly planning for the impacts of failures early in the life of a project can result in catastrophic program or mission disasters.

B. RESEARCH QUESTIONS

Often during the discovery of a problem, the next logical step is to hypothesize or theorize remedies for an established problem. Assuming a specific solution too quickly can present challenges and hinder the true understanding of the problem (Giammarco, Hunt, and Whitcomb 2015). While a plethora of ideas and solutions may be applicable, to address the problem of understanding how failures can impact complex systems, this thesis uses the Monterey Phoenix (MP) modeling software to answer the following research question:

> Can the operational context and derived behavior of UAV swarm systems be analyzed through Monterey Phoenix computational models and simulations to reveal potential failure scenarios, and can failsafe behaviors be generated to counteract the emergence of new failure modes?

The MP modeling environment provides a breakdown of all potential event traces. Given that the research questions call for the revelation of potential failure modes, MP was selected as the modeling environment because it provides a substantial set of results and data, thus allowing the inspection of numerous possible outcomes. Refer to the full description of MP in Chapter II.

II. BACKGROUND AND METHODOLOGY

A. APPROACH TO RESEARCH

1. Monterey Phoenix

To provide an in-depth analysis of every possible permutation in a given scenario, a behavior modeling approach, known as Monterey Phoenix (MP), is used for this failure mode modeling research. MP is "a behavioral model for system and software architecture specification based on event traces" (Farah-Stapleton and Auguston 2013, 271). With explicit objectives and involvement from stakeholders, system architecture modeling is an iterative process to refine the design of a system and ultimately link the requirements to the implementation phases of the system (Auguston 2014). By leveraging MP, this principle may be applied for any process-based system. The view of MP in Figure 1 is of the graph view, which illustrates the ability to see all possible event traces for a given scenario.

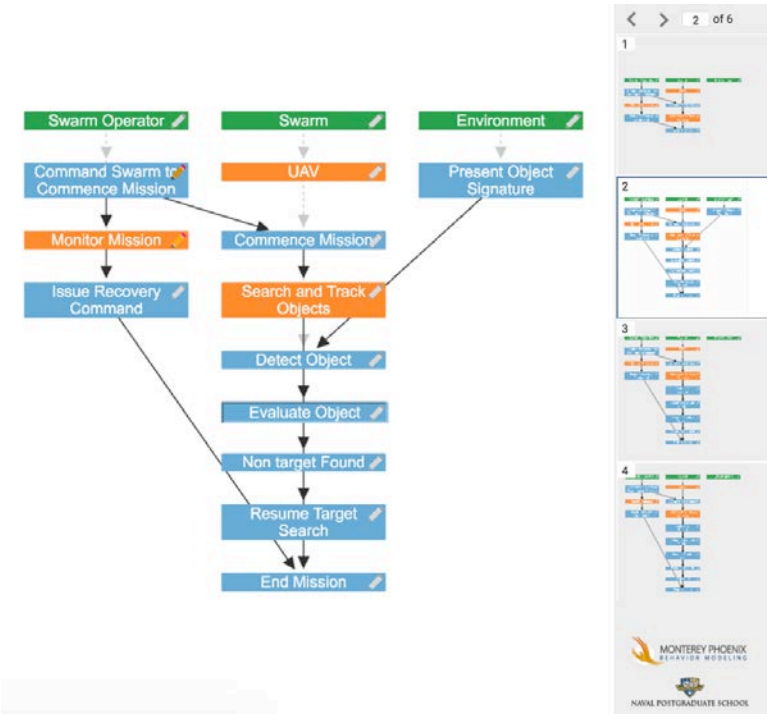


Figure 1. MP Sequence Diagram and Event Trace View

Unlike many modeling approaches, MP provides the capacity to consider all possible actors, or contributors, to a system. This includes contributors like direct and indirect stakeholders, the environment, and other systems. By breaking out the actors as separate entities, a system of systems approach becomes more practical to realize and can be expanded or contracted as needed with little effort or impact to the overall model. Furthermore, MP disconnects the concerns of system behaviors and system interactions, which are traditionally interwoven as hard-coded constraints on multi-actor activity models. These hard-coded constraints are typically limited to random or stochastic simulations that address in contrast only a small subset of potential behaviors.

Monterey Phoenix is in large part a behavior modeling language, but it differs from the traditional functions, components, and connectors by representing the main concepts solely as activities and relationships between activities (Auguston 2016). In his 2016 language manual, Dr. Auguston lists many key concepts; below is a summarization of the concepts relevant to this research.

> A view of the architecture as a high level description of possible system behaviors, emphasizing the behavior of subsystems (components) and interactions between subsystems. MP introduces the concept of event as an abstraction of activity.
>
> The separation of the interaction description from the components behavior is an essential MP feature. It provides for a high level of abstraction and supports the reuse of architectural models. Interactions between activities are modeled using event coordination constructs.
>
> The environment's behavior is an integral part of the system architecture model. MP provides a uniform method for modeling behaviors of the software, hardware, business processes, and other parts of the system.
>
> The event grammar models the behavior as a set of events (event trace) with two basic relations, where the PRECEDES relation captures the dependency abstraction, and the IN relation represents the hierarchical relationship. Since the event trace is a set, additional constraints can be specified using set-theoretical operations and predicate logic. (Auguston 2016)

Specifically, for this research, MP is used to analyze, expose and understand how failure modes impact a swarm. Monterey Phoenix changes the very method by which

system behaviors are modeled by breaking them into discrete, object oriented agent (actor) models made up of possible agent behaviors. This allows for the modeling of failure modes woven into all swarm interactions as separate constraints and computes every permitted permutation of actor behavior with every other possible actor behavior automatically from these models, to generate an exhaustive set of possible scenarios up to a specified scope limit. This automated scenario generation provides a huge set of data for human inspection and assertion checking for the presence or absence of specific failure modes of concern.

In MP, a system's behavior is modeled as a collection of events. As Dr. Auguston states above, there are two relationships that exist between events: precedes and includes. For precedes, if one event must occur before another, a dependency is established. Consider the simple message flow example in Figure 2, where the event of sending a message must precede receipt. For includes, an event may include a subset of events. Again represented in Figure 2, the event "Send" includes "Prepare Message" and "Send Message."

```
SCHEMA Simple_Message_Flow

ROOT Sender:   (* Send *);
ROOT Receiver: (* Receive *);

Send: Prepare_Message Send_Message;

Receive: Receive_Message Read_Message;

COORDINATE  $x: Send    FROM Sender,
            $y: Receive FROM Receiver
    DO ADD $x PRECEDES $y; OD;
```

Figure 2. MP Code – Simple Message Flow Code

Through MP's analyzer tool located at firebird.nps.edu, modelers are able to generate a graphical representation of the possible event traces available from their model. The green blocks represent the root events or actors of the model and the orange and blue blocks correspond to events included in the roots. The orange blocks are

composite events that are made up of blue blocks that correspond to atomic events. The dashed arrows represent inclusion relationships within events and the solid arrows are precedence relationships between events. For example, in Figure 3, the event "Send" is made up of "Prepare Message" and "Send Message." Since both are in the "Send" event, a dashed arrow links "Send" to the nested events.

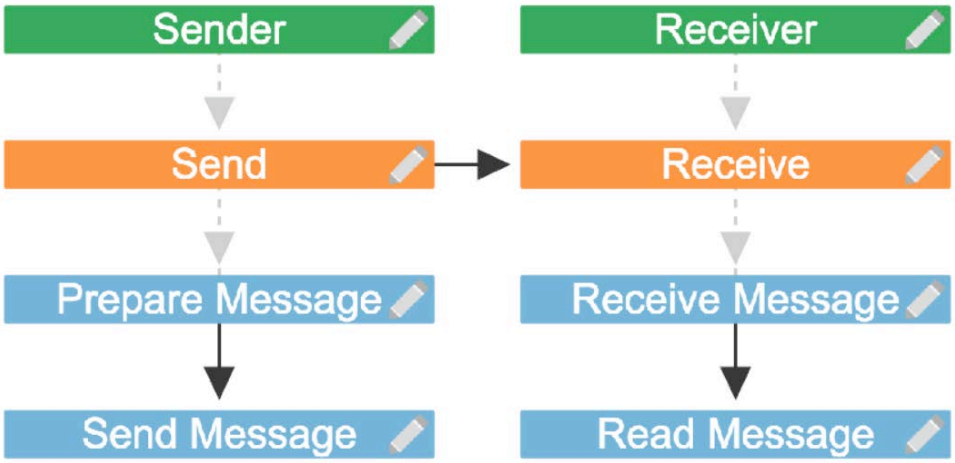


Figure 3. MP Sequence Diagram – Simple Message Flow Model

Monterey Phoenix also provides the ability to represent events in various patterns. Table 1 describes the various patterns in both natural language and MP event grammar. These patterns offer a reference for many of the patterns represented in this thesis research.

Table 1. MP Event Patterns

Pattern Natural Language Description	Pattern Expressed as MP Event Grammar
Ordered sequence of events (B followed by C)	`A: B C;`
Alternative events (B or C)	`A: ( B | C );`
Optional event (B or no event at all)	`A: [ B ];`
Ordered sequence of zero or more events B	`A: (* B *);`
Ordered sequence of one or more events B	`A: (+ B +);`
Unordered set of events B and C (B and C may happen concurrently)	`A: { B, C };`
Unordered set of zero or more events B	`A: {* B *};`
Unordered set of one or more events B	`A: {+ B +};`

Source: Mikhail Auguston, "MP Event Grammar," accessed July 1, 2016, https://wiki.nps.edu/display/MP/Event+Grammar.

a. Small Scope Hypothesis

Given the exhaustive set of scenarios generated, one of the most beneficial features of MP is the ability to check or validate models. Model validation is not a new technique, as it was originally "developed in the 1980s for analyzing protocols and hardware designs that could be expressed as finite state machines" (Jackson 2006). Model checking was proven to be extremely effective in identifying errors, which led to the testing of unbounded system models. This new application introduced the notion of scope considerations. The Small Scope Hypothesis (Jackson 2006) states that most errors can be exposed on small examples or "that systems that fail on large instances almost always fail on small ones with similar properties" and similarities between modeled to actual systems are irrelevant.

Monterey Phoenix generates a finite number of event traces from a model comprising potentially an infinite number of behaviors. By limiting the scope, the same failures can be exposed by simulating only a small number, typically three, of iterations on the model loops. Additionally, the number of event traces and trace generation time

may increase on an exponential scale as the scope limit is increased (Auguston 2016). Consider the following MP example of a simple air-to-air combat scenario between two sets of swarms, each consisting of one or more UAVs, set by the scope limit. Per the model, each UAV can only perform one attack operation, shoot the enemy UAV, which can result in a hit or miss. As a resultant, each UAV can be destroyed or survive and return to its base.

```
SCHEMA Blue_vs_Red
Swarm: {+ UAV +};
UAV: Attack Shoot_enemy_UAV (destroyed | return_to_base);
Shoot_enemy_UAV: (hit | miss);
ROOT Blue: Swarm;
ROOT Red: Swarm;
COORDINATE <!> $h: hit FROM Blue,
           <!> $d: destroyed     FROM Red
    DO ADD $h PRECEDES $d; OD;
COORDINATE <!> $h: hit FROM Red,
           <!> $d: destroyed     FROM Blue
    DO ADD $h PRECEDES $d; OD;
```

Figure 4. MP Code – Swarm vs. Swarm

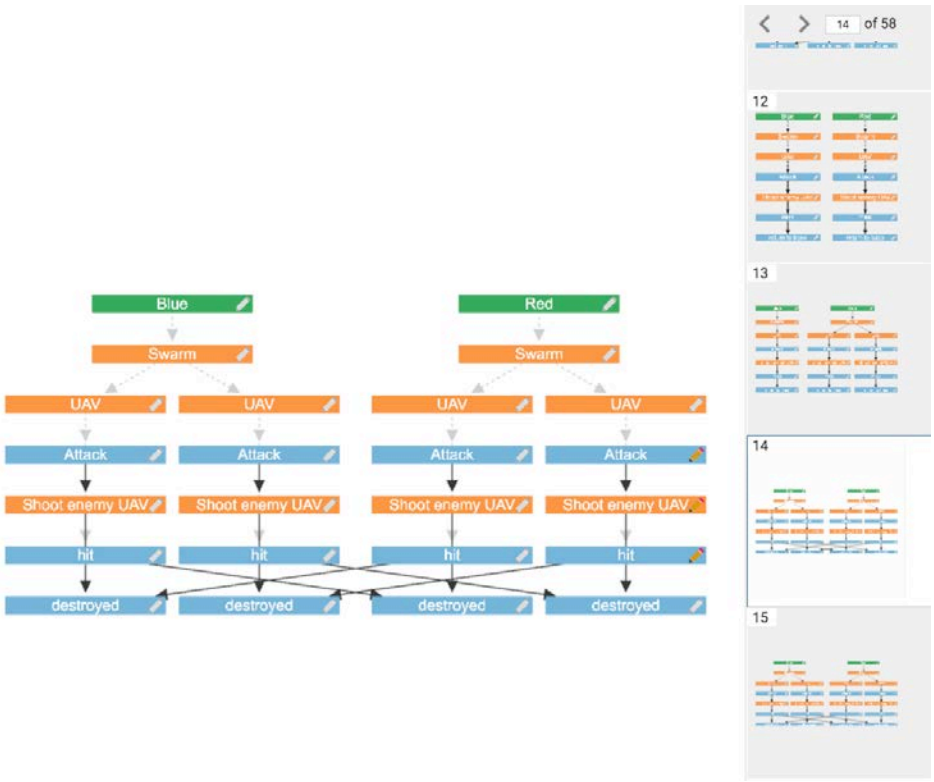


Figure 5. MP Sequence Diagram – Swarm vs. Swarm

When the scope limit is set to one (i.e., each swarm has only one UAV), MP generates a small number of event traces. Like the Punnett square exercise, the possible scenarios are fairly easy to comprehend, which are (a) both UAVs are destroyed, (b) both UAVs survive, (c) the blue UAV survives and the red UAV is destroyed, and (d) the red UAV survives and the blue UAV is destroyed. As the scope limit is increased, the number of event traces grows drastically, see Figure 6. Again, considering the concept of the small scope hypothesis, all scenarios and failure patterns should be identified at or around scope limit three.

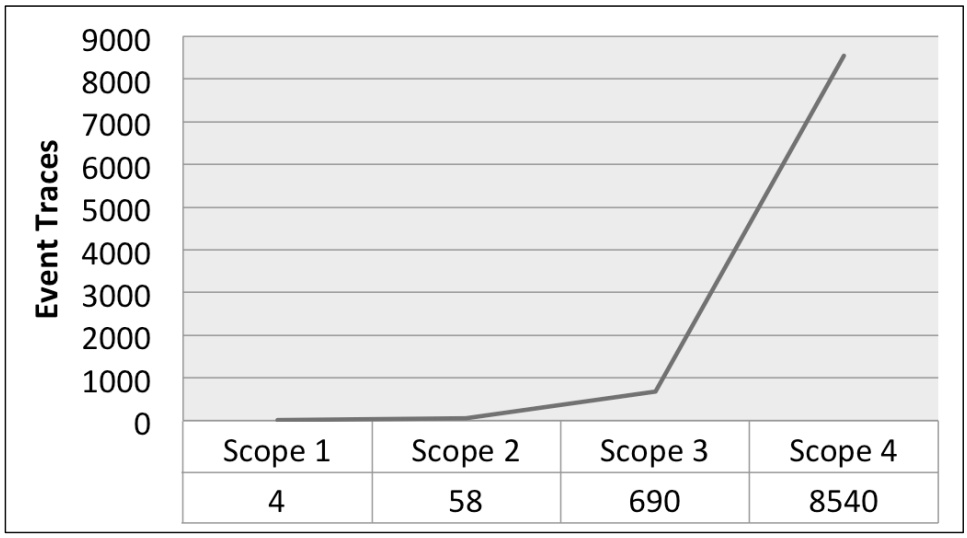


Figure 6. Swarm vs. Swarm Event Trace Graph

Dr. Kristin Giammarco (2012) took the analysis of small scope hypothesis a step further in her dissertation, "Architecture Model-Based Interoperability Assessment," to further test "that most flaws that appear in large instances with many variables and relations also appear in smaller instances (fewer variables and relations)." Using the Alloy Analyzer, her research checked for assertions spanning scopes on a sliding scale from 3 to 20. With an exponential relationship, the number of variables and the time required to solve the possible event traces escalated as the scope increased. The natural assumption would be that confidence levels in the model would increase as the scope increased. Through manual inspection and automated assertion checking, she was able to prove that there were no new behaviors at scope 20 there were not also present at scope 3 (Giammarco 2012).

2. Emergent Behavior

Emergent behavior has received many different definitions across numerous fields of study. According to Dyson, "Emergent behavior is that which cannot be predicted through analysis at any level simpler than that of the system as a whole" (Dyson 1997). He goes on to explain that after everything has been described, emergent behavior is what remains. Consider the UAS operational demonstration at the PACOM Crimson Viper 2010 Exercise, where two emergent behaviors were observed. While a WASP UAV was

executing a demonstrative search and track mission for senior DARPA leaders, a live, large sea hawk tracked, engaged and disabled the WASP.

Figure 7. PACOM Crimson Viper 2010 Exercise. Source: Perschbacher (2010).

The immediate reaction from the project team was disappointment. In attendance was a DARPA Program Manager who had been working to establish a new approach to counter UAS operations. He recognized this event as a desirable type of emergent behavior and subsequently returned to DARPA to create the Falconry – Alternate Lure Training (FALT) project, where live falcons were trained to track and engage WASP UAVs.

The introduction and subsequent rapid growth of unmanned and autonomous systems has had a significant impact on the predictability of the behavior of systems. Furthermore, today's unmanned systems are now part of a system of systems (SoS), wherein not only are engineers and developers required to understand their specific

product, they must now have a firm understanding of how their product fits into a plethora of other systems, systems that are often new and potentially misunderstood.

As a result of this massive growth in unmanned systems development, modeling and simulation (M&S) has become a critical tool in understanding the behavior of these systems. When systems are modeled, they typically have one of two results, expected or unexpected results, but there is often another way of viewing behavior as a subset of expected and unexpected results, which may be considered emergent behavior.

There are many theories and discussions on how emergent behavior is discovered. This research follows the approach described by Gore and Reynolds (2008), where subject-matter experts are introduced early in the concept design "to test a hypothesis about emergent behavior." This approach uses "causal inference procedures to reveal interactions of known abstractions in the model, which will cause emergent behavior. Causal inferencing finds cause and effect relationships among observed variables," which explains or describes a set of observations (Gore and Reynolds, 2008).

The opening example of emergent behavior during the PACOM Crimson Viper 2010 Exercise resulted from an unforeseen interaction with a known environmental element. In order for MP to reveal this emergent behavior, wildlife may need to be modeled as a separate actor that is capable of interacting with other actors. Monterey Phoenix can indirectly help to expose this type of behavior by provoking the MP user's thought process. The mere presence of a general environmental element at a critical step in an event trace may be enough to inspire thoughts about specific examples of environmental elements. The involvement of subject-matter expertise can greatly assist in this process.

B. LITERATURE REVIEW

In her paper presented to the Complex Adaptive Systems Conference, K. Giammarco (2013) describes emergent behavior from a system of systems (SoS) approach using MP. She explains how "the Monterey Phoenix (MP) system architecture modeling framework can be used to structure independent models of task-oriented

systems” followed by the introduction of these independent models into larger paradigms to enable predictions of SoS behavior.

M. Auguston (2014), the developer of Monterey Phoenix, describes the behavior-modeling role the MP framework plays in understanding system architecture and how it aids in establishing a bridge between requirements and implementation.

Similar to the search and rescue (SAR) mission described in later sections, in his Master’s thesis, S. Hunt (2015) establishes a baseline SAR mission for model-based evaluation. This baseline model was evaluated across many modeling techniques, including Monterey Phoenix.

K. Giammarco, M. Auguston, C. Baldwin, J. Crump, and M. Farah-Stapleton (2014) presented the “Controlling Design Complexity with the Monterey Phoenix Approach” paper to the Complex Adaptive Systems Conference, which aimed to “describe how the Monterey Phoenix (MP) approach can be used to decompose a complex problem into smaller, more manageable models.”

J. Pilcher (2015) analyzes MP as a tool to aid the Department of Defense Architecture Framework (DODAF) to better understand and intercept errors earlier in the design process. Interestingly, through her order-processing model, she was able to expose an unwanted or unexpected emergent outcome (scenario ending in a waiting state) through the MP model (Pilcher 2015).

V. Steward (2015) analyzed and compared two modeling tools, Innoslate and MP, for a search and rescue (SAR) mission. During her research, she identified an emergent behavior using MP, where a UAV returned to base too soon and did not search for additional survivors or wreckage once one was found.

The knowledge gathered from this literature review provides a foundation for the research performed in this thesis. The two examples of emergent behavior lay the groundwork for exposing unwanted emergent behaviors in UAV swarm use cases.

C. SWARM DEFINED

Minar, Burkhart, Langton, and Askenazi (1996) define a swarm as "a collection of agents with a schedule of events over those agents" (Minar, Burkhart, Langton, and Askenazi 1996). While there is no mention of the number of agents, the term "swarm" implies there is more than one agent. As it relates to the military, the term swarm is usually applied to a strategic battle operation through a purposefully arranged and organized set of assets with the intent to attack from every direction. Whether near or far away from the enemy line, this overwhelming approach is accomplished with the defensible "pulsing" of navigable entities (Arquilla and Ronfeldt 2013). Unmanned aerial vehicle systems operating as a swarm have useful applications in a myriad of possible scenarios on and off the battlefield.

In this research, the focus is placed on the utilization of a group of UAVs organized in a mesh-like structure to divide the workload of a search and rescue (SAR) mission (Figure 8). Swarm-based SAR missions can be worked more rapidly by blanketing more of the search area when the collection is spread out. Through redundancy, swarms can increase the chances of success mission execution. If a UAV drops out or crashes because of mechanical failure, the rest of the group can take over and complete the mission (Frantz 2005).

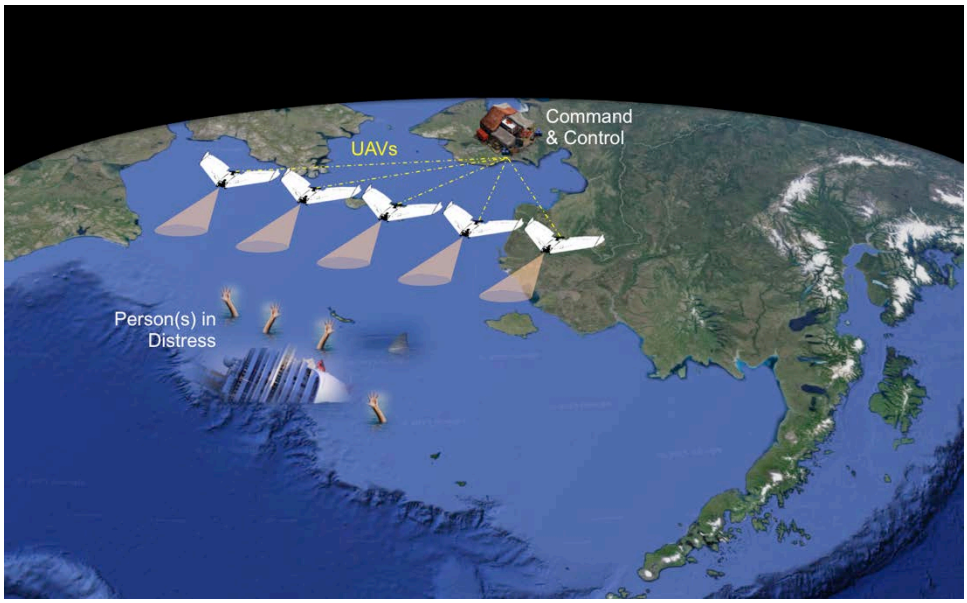


Figure 8. SAR Operational View

The traditional definition of a swarm, as discussed above, does not necessarily apply to the MP modeling approach used in this research, which allows for the scope of the swarm size to be as small as 1 UAV. Therefore, due to this scope-based modeling approach defined in the small scope hypothesis section, the number of UAVs in a swarm is comprised of one or more assets.

THIS PAGE INTENTIONALLY LEFT BLANK

III. THE SEARCH AND RESCUE MODEL

Swarm UAV missions encompass simultaneously flying UAVs in autonomous and coordinated flight modes, such as waypoint following or sensor-based navigation. To adequately test and analyze findings during model evaluation a baseline or design reference mission (DRM) model is created. Specifically for this thesis, a swarm-based search and rescue (SAR) mission is established. The concept of the SAR mission is familiar, which enables the focus to be on the recognition of emergent behaviors.

A. APPLICABLE UAV FOR SAR MISSION

From fixed-wing to multi-rotor, there are many different types of UAVs that can accomplish a SAR mission. The behaviors, failure modes and corresponding failsafe behaviors can be considerably different for each airframe. Consider the failure mode of a low battery or fuel for a fixed-wing and a quad-rotor UAV. A potential failsafe behavior for the fixed-wing aircraft may be to reduce throttle and glide until it is safe to land. For a quad-rotor, or any multi-rotor aircraft, gliding is not an option. Thus, a solution-specific approach for failure modes cannot be universal for all UAV types. For the purposes of this research, the Zephyr II UAV will be considered in all modeling scenarios. See Table 2 for specific aircraft specifications and characteristics.

Table 2. Zephyr II – UAS Data

Feature	Value
UAS Name	Zephyr II
Manufacturer	Rite WingRC
Wing Span	56 in.
Length	15.5 in.
Dry Weight	5.5 lbs.
Max Gross Weight	7.0 lbs. *(1.5 lbs. payload capacity)*
Max Speed	40 m/s
Cruise Speed	18 m/s
Stall Speed	10 m/s
Endurance	60 minutes (*May vary, depends on type of batteries used)*
Engine	Brushless 1300 kV or similar
Fuel	Two – Four 11.1V LiPo batteries *(connected in parallel)*
Max Range	40 km (*May vary, depends on type of batteries used)*
Communications	2.4 – 2.48 GHz, up to 30 Mbps 902 – 928 MHz, up to 115.2 kbps
Default Payloads	Mesh network router PC104 or ODroid computer GoPro Camera

Source: Michael Day, Naval Postgraduate School (personal communication, 2014)

Dr. Timothy Chung (2014) explains that each Zephyr II aircraft has a primary flight control systems that is managed by an autopilot module known as ArduPilot. ArduPilot is separate from the swarming control system developed by NPS. Each aircraft is an independent agent in the swarm, and is required to synchronize with other swarming agents: "Failsafe behaviors on agents in a swarm are managed the same way as "single plane sorties" (Chung 2014).

B. ACTORS

When modeling swarm-based SAR missions, the capacity exists for a large number of agents or actors operating both individually and as a group within the model. The following list describes a potential set of actors for a conceptual swarm-based SAR mission.

- Flight Crew – The Flight Crew is an encompassing term that refers to the personnel necessary to manage all logistical aspects required during a SAR mission.
- Swarm – The Swarm is an encompassing term, but only refers to the finite collection UAVs assigned to the active SAR mission.
- Mission Commander – The Mission Commander is the individual in charge of all swarm operations for the SAR mission.
- Swarm Operator – The Swarm Operator works very closely with the Mission Commander and is the individual responsible for piloting the swarm. While still required, this position can become marginal as the size of the swarm grows. Large quantities of UAV must operate with a high level of autonomy and "increasing the autonomy of unmanned platforms could reduce the number of operators per vehicle, thus simplifying the task of the operating crew controlling vehicles involved in complex missions and potentially reducing costs" (Rabbath, Alain, and Léchevin 2010).
- Range Control – Range Control receives a briefing on mission planning and is notified of launch and landing attempts
- Safety Coordinator – The Safety Coordinator executes prefight checklists, verifies the environmental conditions and provides valuable information to the Mission Commander.
- Physical Environment – The Physical Environment encompasses all external factors that are outside of the control of machine and human actors. While this is not an obvious actor, it plays a vital role in modeling behaviors of systems. Chances of receiving inaccurate results are greatly increased if this is not included.
- Person in Distress (PID) – A PID is any single or group of individuals that require aid through the SAR mission.

The interactions between each actor, which can be interactions among many, can quickly become detailed and complex. Even at the smallest subroutines of a SAR swarming mission scenario, the potential number of exchanges can be overwhelming. Dr.

Chung explains that flight plans for all simultaneously operating aircraft are reviewed prior to takeoff and confirmed by Swarm Operator and Mission Commander. Unmanned aerial vehicles communicate wirelessly with each other, subject to constraints of LOS and range, to share information like telemetry and sensor data to facilitate use of coordination algorithms. Compact telemetry data is provided to the Swarm Operator for situational awareness of all aircraft (Chung 2014). The activities are just a few behaviors performed by a large collection of actors. Figures 9 and 10 illustrate some of the potential subroutines of a SAR mission scenario.

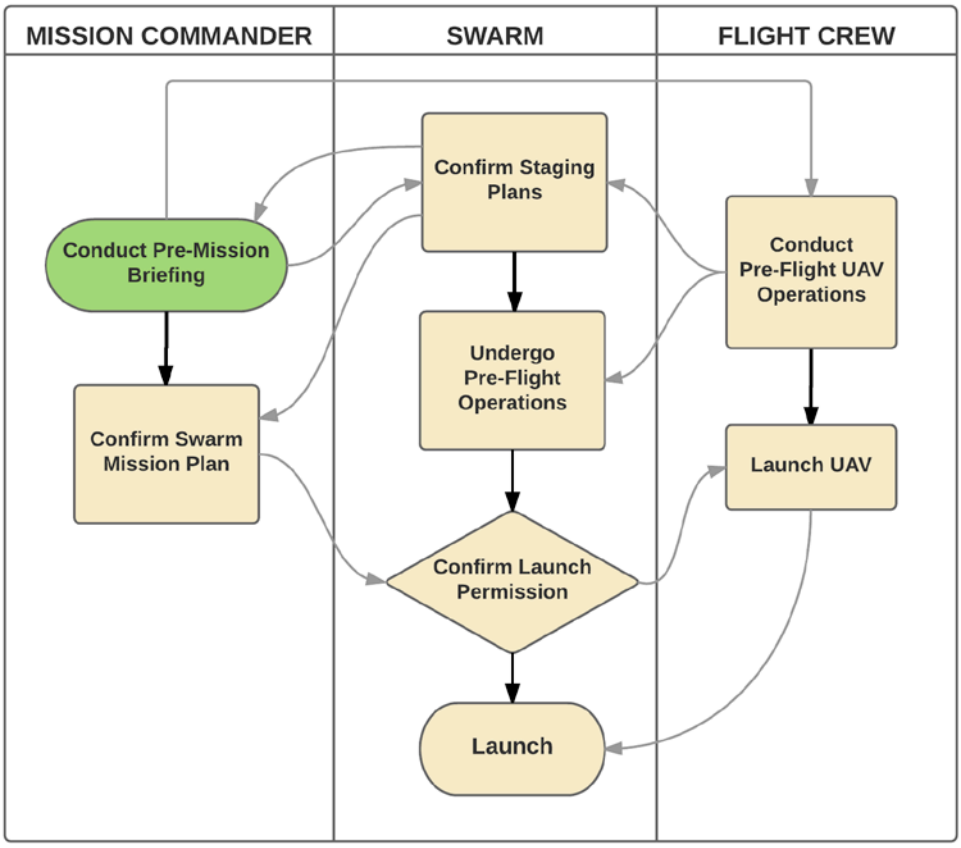


Figure 9. SAR Pre-mission Startup Operations

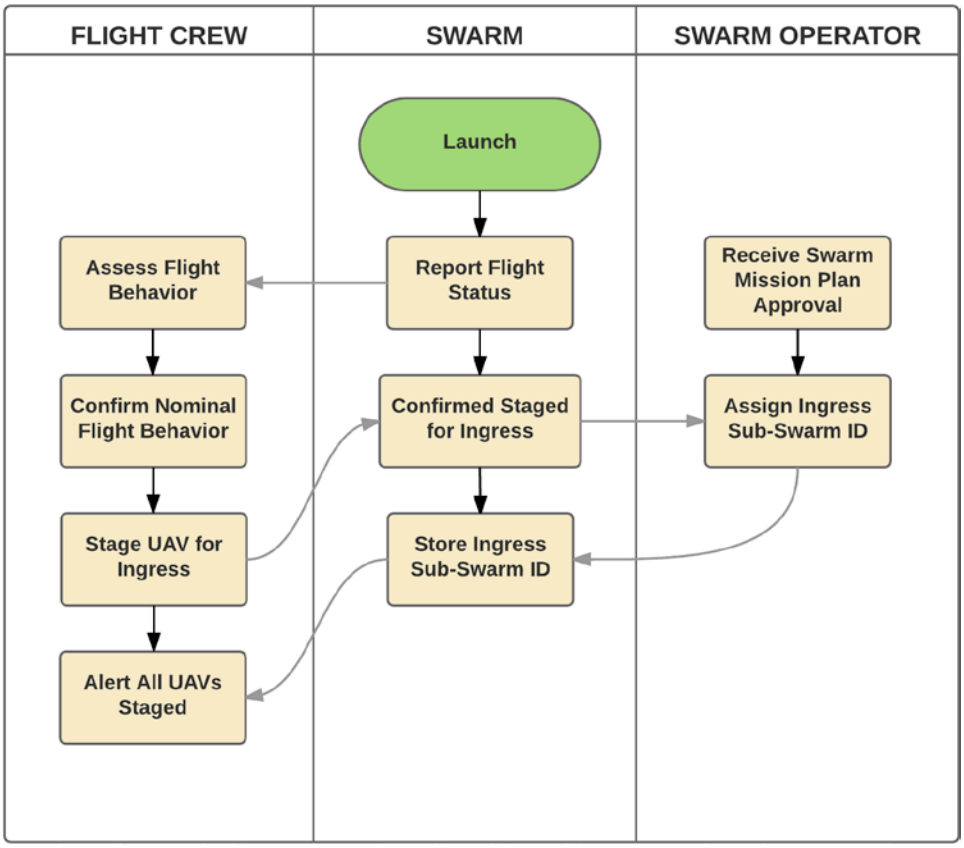


Figure 10. SAR Initial Launch

C. THE COMPLETE DESIGN REFERENCE MISSION

When these interactions are modeled together to form a complete SAR mission, an intricate set of events and interactions between and across multiple actors begin to appear. As illustrated in Figure 11, the Mission Commander and Swarm Operator have multiple interactions with almost all actors in the model.

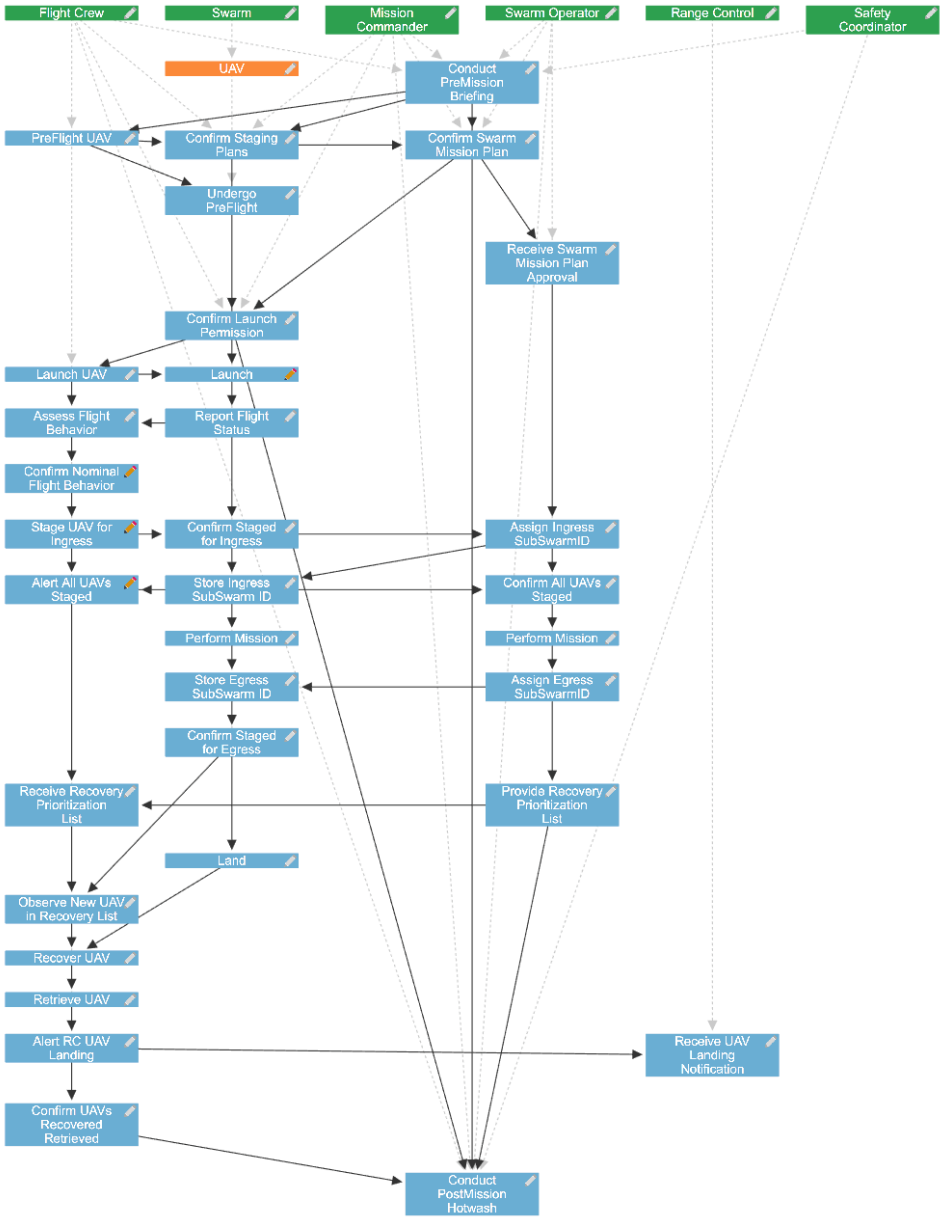


Figure 11. MP Sequence Diagram – Swarm SAR CONOPS

1. Scaled-Down Approach to Failure Mode Research

Inserting failure modes directly into the complex SAR mission referenced in the previous section can prove to be futile. The number of interactions can quickly overwhelm the desired outcome of learning from failure mode insertion.

Alternatively, a more useful approach is scale the model down to key interactions that allow for the seamless insertion of failure modes. This does not mean the model becomes less representative of the desired event traces. A system of systems analysis approach can aid in producing a succinct and focused model capable of accepting relevant failure modes. As discovered through various modeling iterations, the author considered three main approaches to scaling down the model through various modeling iterations: isolated model extraction, phased model extraction, and model abstraction.

Isolated model extraction is selecting a specific subsystem within a model and performing analyses on that single subsystem. This approach can be especially useful when the interactions and behaviors being observed have little to no impact on any linked systems. Through examination and discovery, patterns active at the system level, are repeated at the subsystem level and in successive layers until all common information is extracted. Therefore, the modeled system in MP contains a distinct layer of structure and function (Garcia 2015). Model extraction can present major issues, as it is difficult to isolate a failure mode to a single subsystem. When employing model extraction as an approach to scaling down a model, it is paramount that a thorough trace study be performed to determine trickle-down impacts to other systems.

Similar to model extraction is a phased extraction approach, where an activity or process is broken down into phases before any models are actually executed. Each phase is treated and analyzed independently for failure modes unique to the respective phase. This approach drastically reduces the number of event traces that would normally be of no interest to other phases in the sequence. If the same behavior is present in more than one phase, that behavior will need to be represented across each applicable phase instance. This requires significant planning and decomposition to ensure incorrect assumptions are not made during the modeling process.

Used primarily in this research, the model abstraction approach attempts first to break a model down into its major interactions. Once these interactions are determined, the detail for each interaction is reduced. “Abstracted systems are simplifications of more complex dynamical systems that retain some important information, such as controllability, about the original system” (Mellodge and Pushkin 2008). For example,

the SAR model in Figure 12 is a scaled down version of the complete DRM featured in the previous section. This simplified representation allows for the streamlined or algorithm-based introduction of failure modes. Upon execution, MP presents every event trace combination, making it possible quickly and easily to identify scenarios where a particular failure mode may influence a system. If a model is too complex, as in the complete SAR DRM, failure modes may become hard to discern.

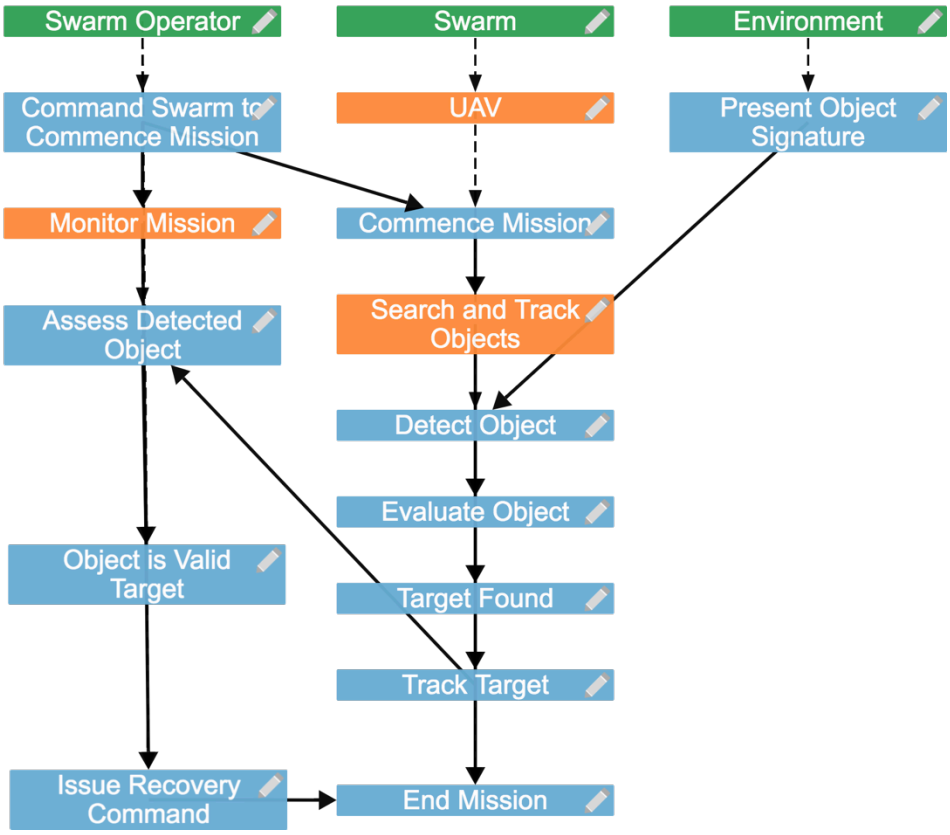


Figure 12. MP Sequence Diagram – Abstracted SAR Model

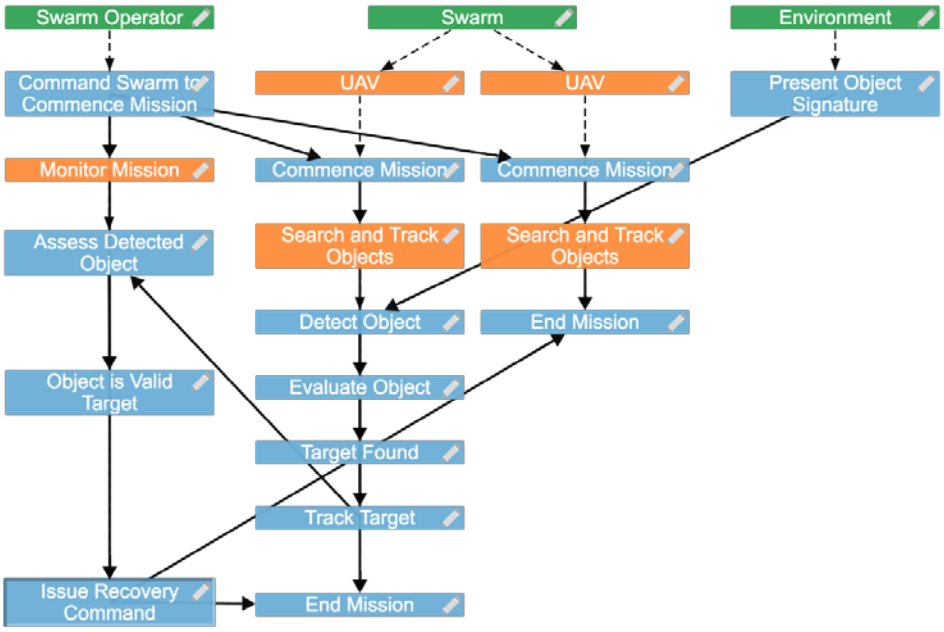


Figure 13. MP Sequence Diagram - Abstracted SAR Model (Scope = 2)

In Figure 13, the scope is changed to 2, which illustrates how MP is capable of modeling a number of assets and separate missions on a sliding scale. In this event trace example, one UAV does not find any targets to analyze and subsequently ends the mission, while the other UAV finds and tracks a valid target.

2. Monterey Phoenix Version of SAR

Monterey Phoenix "represents an event as an abstraction of an activity, the behavior of a system can be modeled as a set of events with two binary relations defined for them" (Farah-Stapleton and Auguston 2013, 273) and uses an event grammar that allows for the compact specification of behavior for each component (Giammarco and Auguston, 2013). This binary relationship is represented in Figure 13 where the swarm operator has the monitor mission task that is composed of a subroutine. When called, this subroutine requires the swarm operator to assess a detected object, followed by the binary decision of verifying the object's validity. This binary decision process aids in the failure mode and failsafe behavior analysis discussed in the next chapter.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
    DO COORDINATE <!> $b: Commence_Mission       FROM Swarm
        DO ADD $a PRECEDES $b; OD;
    OD;

COORDINATE <!> $a: Track_Target FROM Swarm,
    $b: Assess_Detected_Object  FROM Swarm_Operator
        DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command   FROM Swarm_Operator
    DO COORDINATE <!> $b: End_Mission   FROM Swarm
        DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
               $b: Detect_Object            FROM Swarm
    DO ADD $a PRECEDES $b; OD;
```

Figure 14. MP Code – Baseline SAR Mission

IV. FAILURE MODES AND FAILSAFE BEHAVIORS

This chapter presents a description and analysis of many potential SAR UAS failure modes and corresponding failsafe behaviors using the Monterey Phoenix (MP) modeling tool. It is important to note, failsafe behaviors do not always link back to the source failure mode in a one-to-one ratio. Often, failsafe behaviors can be mission driven, which means the failsafe behavior for one mission (e.g., SAR) may not be applicable to a failure mode in a different mission. For the purposes of this research, failsafe behaviors are specific to Swarming SAR missions. The baseline SAR model does not produce a substantial number of event traces as the scope is increased. This is partially caused by the high level model chosen for analysis and to aid in understanding the failure modes in a somewhat isolated approach of breaking each failure mode out by itself.

A. FAILURE MODES

1. Autopilot Failure

Many UAV missions involve some characteristic of waypoint navigation, which is the preprogrammed or live planning and deliberate guidance of a UAV's path during flight. An autopilot failure mode is an electronics failure that occurs if the UAV fails to adhere to navigation planning, primarily during waypoint navigation.

Failsafe behavior: In a swarm environment, an autopilot failure can have devastating impacts to the entire swarm. Depending on the proximity of nearby assets, the failing UAV can collide with other UAVs, enter into prohibited territories, or create a safety hazard for nearby personnel. Therefore, the optimal failsafe behavior for an autopilot failure scenario is to first attempt to take over manual control if the support is present. If that is not feasible, the UAV should immediately reduce altitude and land. The landing method is a mission driven behavior that does not warrant decomposition at this level of abstraction. Refer to Figures 15 and 16 for the MP codes used and an example event trace generated from the model.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  ( Detect_Object
    Evaluate_Object
      ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Autopilot_Failure
    /*Failsafe Behavior*/
    Attempt_Manual_Control
    (
      Manual_Control_Success Return_to_Launch |
      Manual_Control_Fail Controlled_Crash
    )

  )
*);

COORDINATE  $a: Command_Swarm_to_Commence_Mission   FROM Swarm_Operator
     DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD; /* 10b */

COORDINATE  <!> $a: Track_Target    FROM Swarm,
        $b: Assess_Detected_Object  FROM Swarm_Operator
    DO ADD $a PRECEDES $b; OD;

COORDINATE  $a: Issue_Recovery_Command      FROM Swarm_Operator
     DO  COORDINATE <!> $b: End_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!>  $a: Present_Object_Signature    FROM Environment,
        $b: Detect_Object               FROM Swarm
        DO ADD $a PRECEDES $b; OD;|
```

Figure 15. MP Code – Autopilot Failure

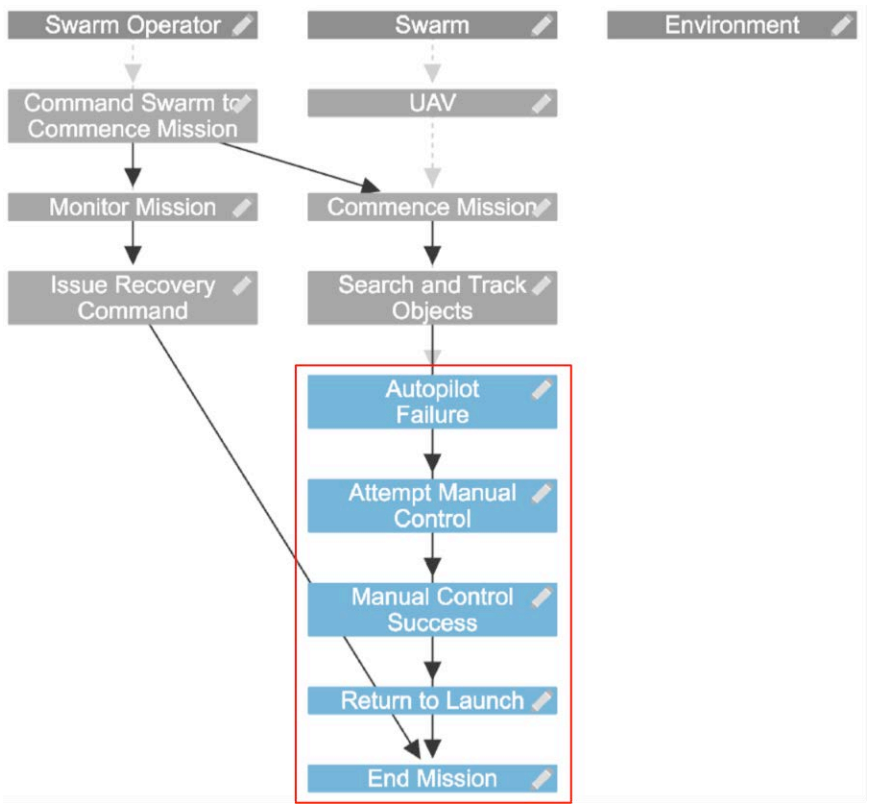


Figure 16. MP Sequence Diagram – Autopilot Failure

As displayed in the sequence diagram in Figure 17, MP affords the ability to adjust the size of the search team or UAVs in this case. Despite that one UAV enters into an autopilot failure mode, all relevant actors are still traced to their cascaded behaviors and other actors. For example, the swarm operator is responsible for monitoring the mission of all active assets. This includes checking for the validity of detected targets and issuing track or recovery commands, even if they are to occur simultaneously. Therefore, in this example, when the failed UAV is given the command to return to launch, MP continues to trace the other responsibilities of the swarm operator.

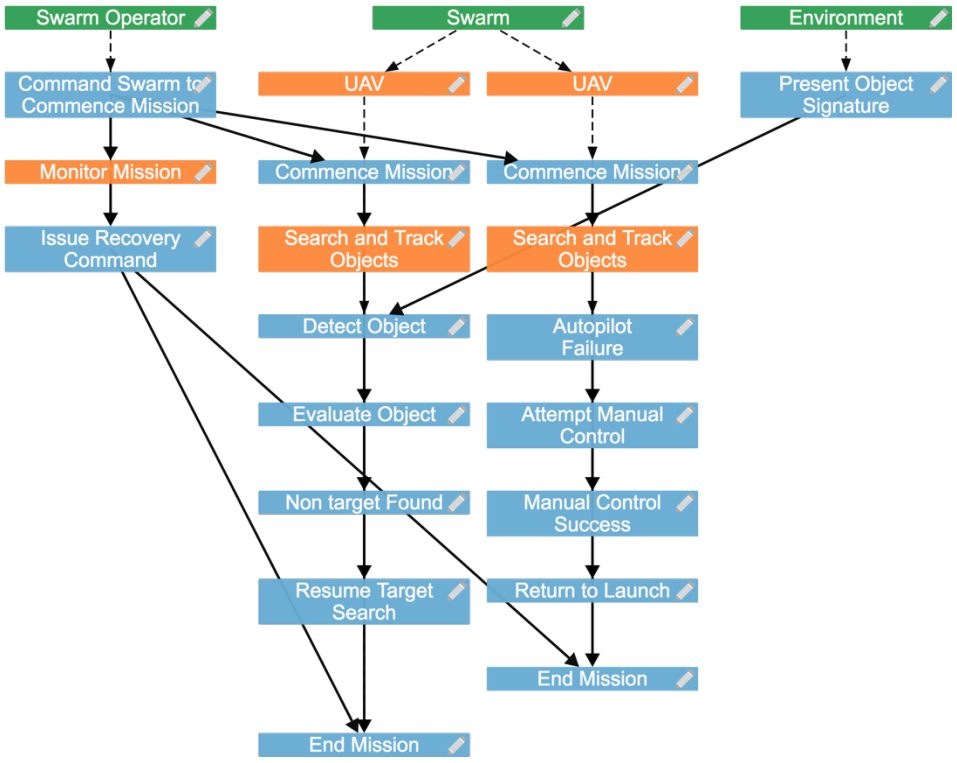


Figure 17. MP Sequence Diagram – Autopilot Failure (Scope = 2)

2. Bingo Fuel

Whether powered by a gas or onboard batteries, UAVs are and will always be limited by the inherent range of their fuel source. “Bingo fuel” is the calculation of the point at which there is just enough fuel to make it to the pre-designated landing location, which can be the original launch or some other rendezvous location. Some may not consider this a failure mode in the traditional sense, as failure modes are typically unanticipated scenarios. While bingo fuel can and should be a predicted behavior, for the purposes of this research, a bingo fuel failure mode is considered an abnormal event that occurs long before the anticipated bingo fuel range. For example, the Zephyr II has a max range of 40 km and an endurance of 60 minutes. If 20 minutes into the flight the Zephyr II enters bingo fuel, this would be considered an unanticipated and abnormal event that would need to be mitigated through a appropriate failsafe behavior.

Failsafe behavior: The assumption in a bingo fuel situation is that the operator or autonomous system still has complete command and control over flight operations. This

failure mode characteristic allows for great flexibility in failsafe options. The most common failsafe behavior for bingo fuel is return to launch (RTL), a predetermined landing area or autonomously land.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  ( Detect_Object
    Evaluate_Object
      ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |
    /* Failure Mode */
    Bingo_Fuel
    /* Failsafe Behavior */
    Check_Fuel_Level
    (
      Is_Returnable Return_to_Launch |
      Is_Not_Returnable Auto_Land
    )
  )
*);

COORDINATE  $a: Command_Swarm_to_Commence_Mission   FROM Swarm_Operator
     DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
          DO ADD $a PRECEDES $b; OD;
    OD; /* 10b */

COORDINATE  <!> $a: Track_Target    FROM Swarm,
        $b: Assess_Detected_Object  FROM Swarm_Operator
    DO ADD $a PRECEDES $b; OD;

COORDINATE  $a: Issue_Recovery_Command      FROM Swarm_Operator
     DO  COORDINATE <!> $b: End_Mission     FROM Swarm
          DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!>  $a: Present_Object_Signature    FROM Environment,
        $b: Detect_Object                   FROM Swarm
        DO ADD $a PRECEDES $b; OD;
```

Figure 18. MP Code – Bingo Fuel

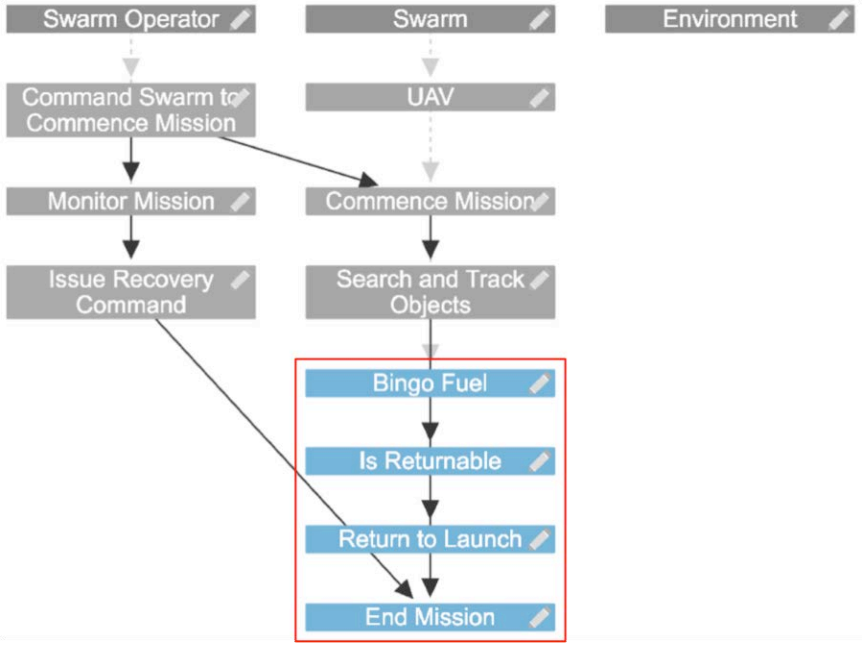


Figure 19. MP Sequence Diagram – Bingo Fuel

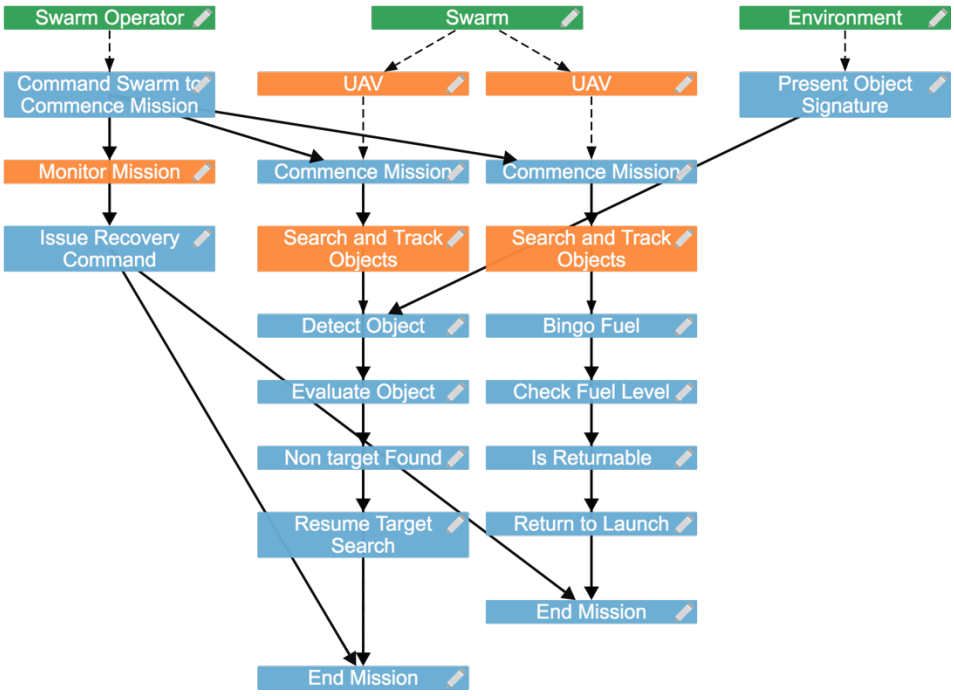


Figure 20. MP Sequence Diagram – Bingo Fuel (Scope = 2)

3. Control Surface Malfunction

There are many critical systems like motors, sensors and onboard computers that enable a UAV to take flight. All of these components are inconsequential without the actuators and control surfaces. A control surface malfunction is a mechanical failure impacting UAV actuators like ailerons, elevators, rudders and propellers.

Failsafe behavior: The unfortunate reality about control surface failures is the operator does not have many options as it relates to failsafe behaviors. There are mitigation techniques that can be employed to avoid complete loss of the air vehicle. Many of these require rebooting the electronics that control the failed surface component, fully actuating the failed surface component in both abduction and adduction extremes and a combination of both. The following MP code and corresponding graph in Figures 21 and 22 executes a tiered approach where multiple failsafe behaviors are tried before a controlled crash is attempted. Given the compound implications of this failure, even if the failure is mitigated through either of the failsafe behaviors, the UAV is directed to return to launch (RTL) for evaluation.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  ( Detect_Object
    Evaluate_Object
      ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |
    /*Failure Mode*/
    Control_Surface_Failure
    /*Failsafe Behavior*/
    Controller_Reboot
    (
      ( Failure_Remains
        Component_Actuation
        ( Failure_Remains Attempt_Controlled_Crash |
          Failure_Mitigated Return_to_Launch
        )
      ) |
      Failure_Mitigated Return_to_Launch
    )
  )
*);

COORDINATE  $a: Command_Swarm_to_Commence_Mission   FROM Swarm_Operator
     DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
          DO ADD $a PRECEDES $b; OD;
    OD; /* 10b */

COORDINATE  <!> $a: Track_Target    FROM Swarm,
        $b: Assess_Detected_Object  FROM Swarm_Operator
    DO ADD $a PRECEDES $b; OD;

COORDINATE  $a: Issue_Recovery_Command      FROM Swarm_Operator
     DO  COORDINATE <!> $b: End_Mission     FROM Swarm
          DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!>  $a: Present_Object_Signature    FROM Environment,
        $b: Detect_Object               FROM Swarm
        DO ADD $a PRECEDES $b; OD;|
```

Figure 21. MP Code – Control Surface Failure

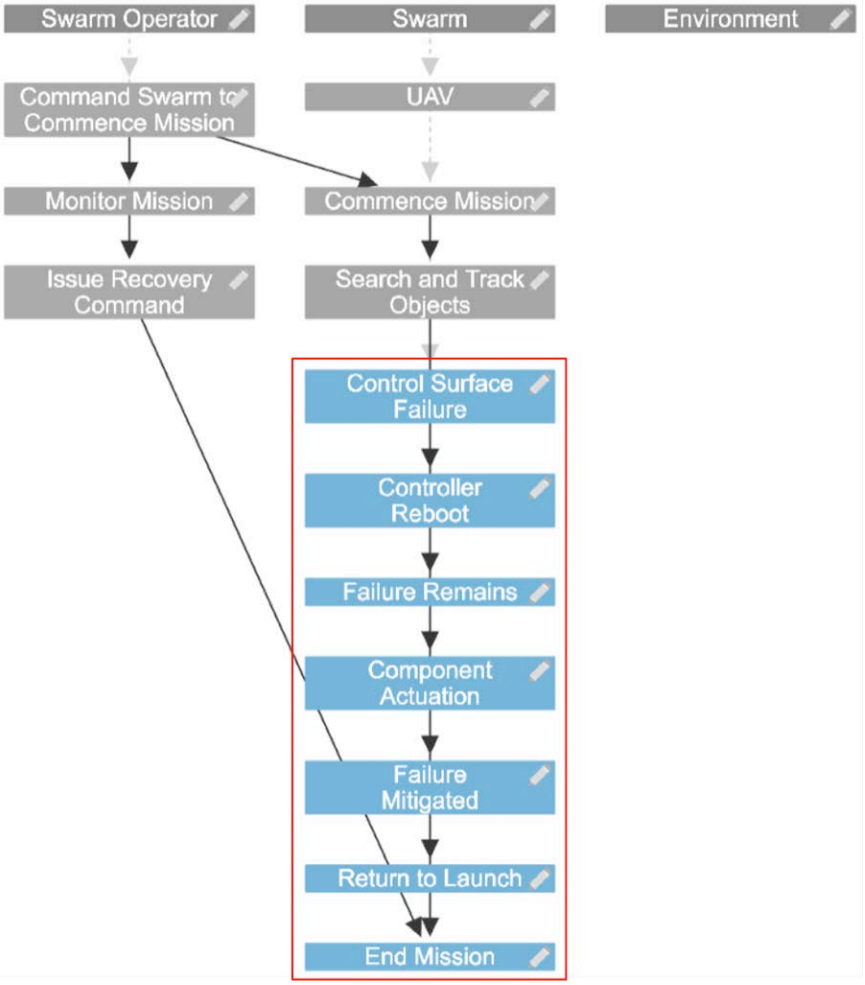


Figure 22. MP Sequence Diagram – Control Surface Failure

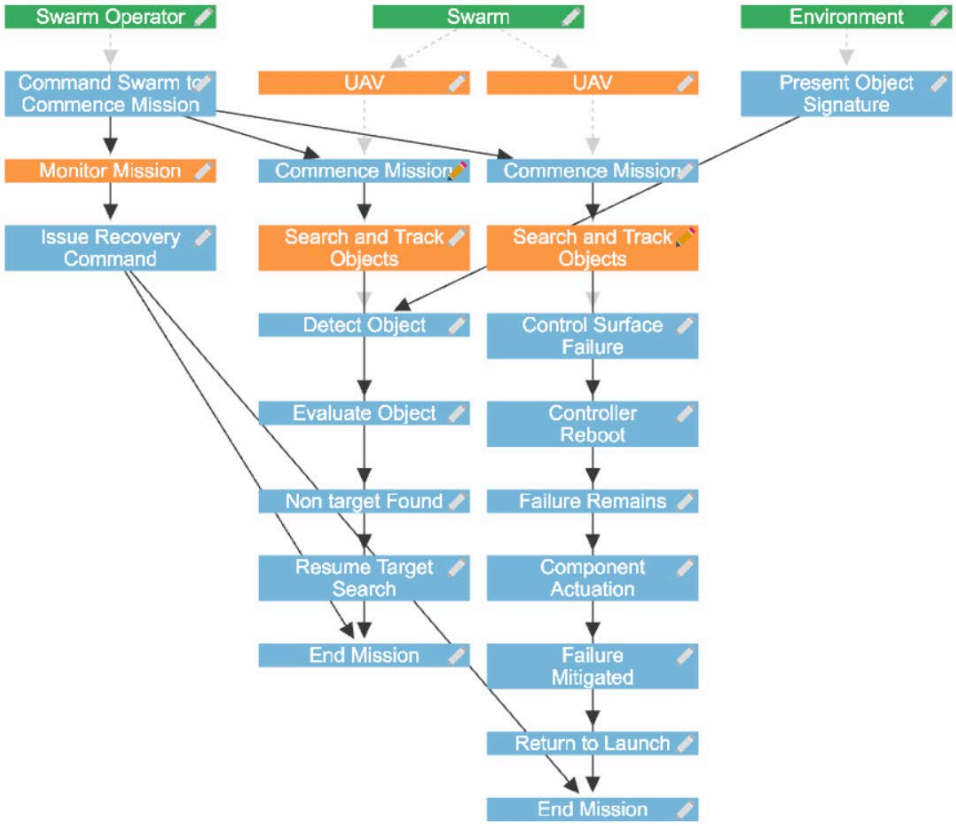


Figure 23. MP Sequence Diagram – Control Surface Failure (Scope = 2)

4. Ground Control Station (GCS) Loss of Link

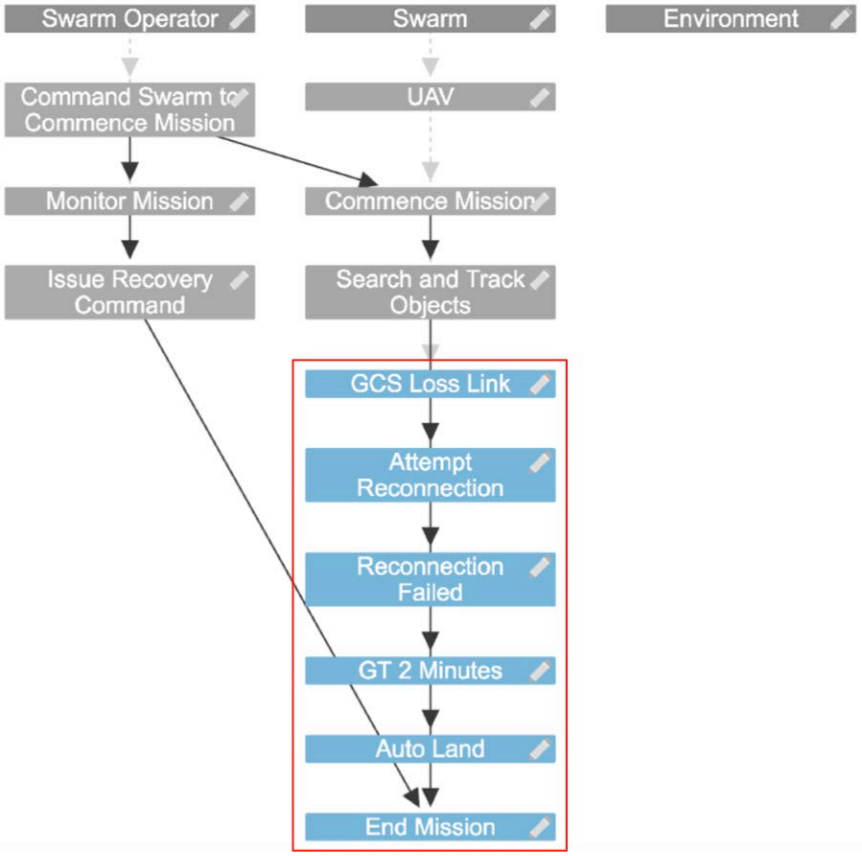
"Drones are dependent on wireless transmissions to relay commands and navigational information. Those connections can be fragile. Records show that links were disrupted or lost in more than a quarter of the worst crashes" (Whitlock, 2014). For a swarm mission scenario, a GCS is used heavily in centralized and hybrid command and control setups. When a loss of link occurs, the swarm operator loses the ability to direct the swarm. The introduction of autonomy into swarms has helped mitigate this risk, but there will almost always be a need for operators to have some supervised autonomy control over the swarm, which will require a GCS or some other medium. For the purposes of this research, a 1:1 GCS to UAV ratio is used when modeling the failure mode.

Failsafe Behavior: With many failure modes, the UAV should attempt some auto-correction behaviors before assuming the failure is irreversible, which is the case with a GCS loss of link scenario. For example, when a UAV loses its link to the GCS, a recursive check for loss linkage may occur for a predetermined N minute timeframe (e.g., two minutes). If the timeframe expires and the GCS link has not reestablished connection, the failsafe behavior is enacted.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  ( Detect_Object
    Evaluate_Object
      ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |
    /*Failure Mode*/
    GCS_Loss_Link
    /*Failsafe Behavior*/
    Attempt_Reconnection
    (
      (Reconnection_Failed (LT_2_Minutes | GT_2_Minutes Auto_Land)) |
      (Reconnection_Successful Resume_Target_Search)
    )
  )
*);

COORDINATE  $a: Command_Swarm_to_Commence_Mission   FROM Swarm_Operator
     DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD; /* 10b */

COORDINATE  <!> $a: Track_Target    FROM Swarm,
        $b: Assess_Detected_Object  FROM Swarm_Operator
    DO ADD $a PRECEDES $b; OD;

COORDINATE  $a: Issue_Recovery_Command      FROM Swarm_Operator
     DO  COORDINATE <!> $b: End_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!>  $a: Present_Object_Signature    FROM Environment,
        $b: Detect_Object               FROM Swarm
        DO ADD $a PRECEDES $b; OD;
```

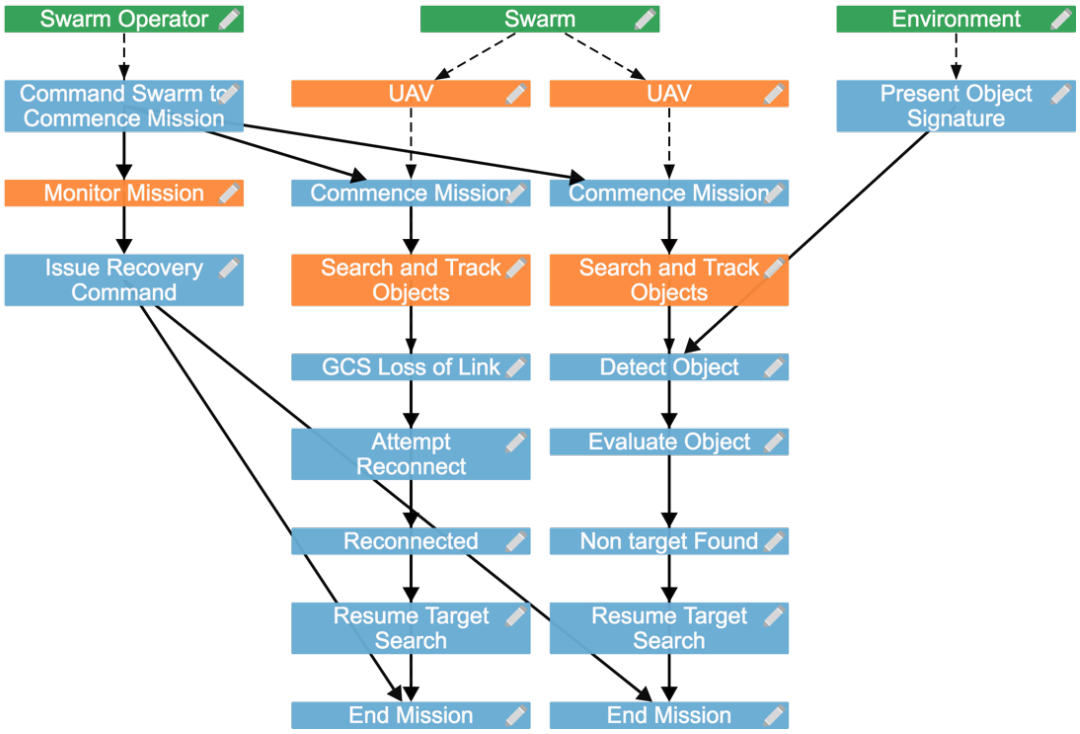
LT = less than and GT = greater than

Figure 24. MP Code – GCS Loss of Link



LT = less than and GT = greater than

Figure 25. MP Sequence Diagram – GCS Loss of Link



LT = less than and GT = greater than

Figure 26. MP Sequence Diagram – GCS Loss of Link (Scope = 2)

5. Loss of Global Positioning System (GPS)

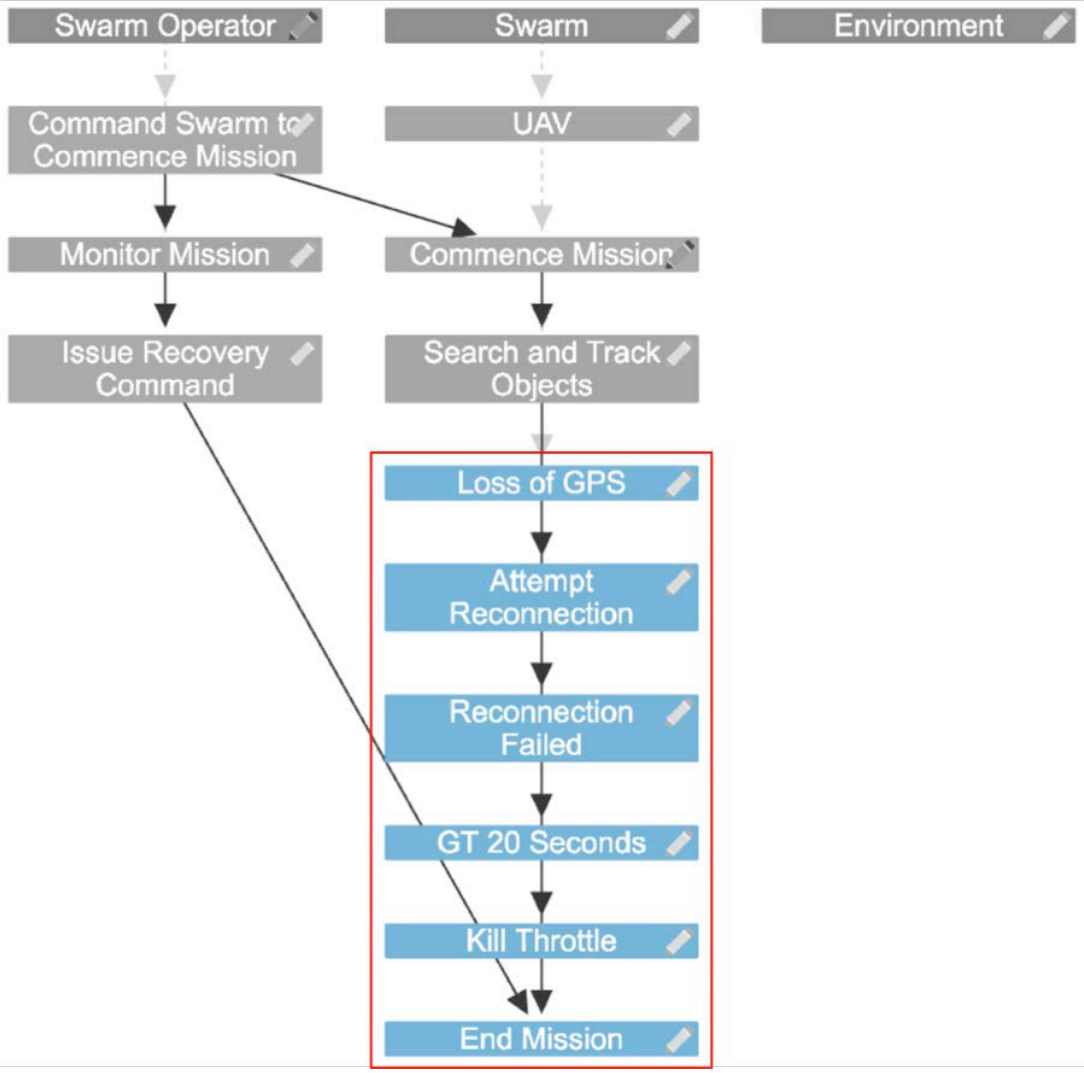
Aerial vehicles have relied on GPS since its introduction. GPS provides both the operator and UAV with extremely accurate information on location. When a UAV loses the ability to use GPS, it becomes unaware of its coordinate-based location in the sky, which "result in inaccurate positions, and can have important consequences in dense urban terrain" (Rabbath, Alain, and Léchevin 2010) and other heavily populated areas.

Failsafe Behavior: Losing GPS is handled much like the GCS loss of link failure mode. When losing GPS, the UAV should also attempt to reconnect, but on a much shorter timetable. An abbreviated timetable (e.g., 20 seconds) is required due the safety implications of losing GPS. Depending on the speed and direction, without GPS the UAV can quickly waver into high-risk territories. Once this timetable has expired, the UAV should kill the throttle, forcing a controlled crash. Furthermore, if GPS connection is reestablished, the UAV should immediately return to launch for evaluation.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  ( Detect_Object
    Evaluate_Object
      ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |
    /*Failure Mode*/
    Loss_of_GPS
    /*Failsafe Behavior*/
    Attempt_Reconnection
    (
      (Reconnection_Failed (LT_20_Seconds | GT_20_Seconds Kill_Throttle)) |
      (Reconnection_Successful Return_to_Launch)
    )

  )
*);

COORDINATE  $a: Command_Swarm_to_Commence_Mission   FROM Swarm_Operator
     DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD; /* 10b */

COORDINATE  <!> $a: Track_Target    FROM Swarm,
        $b: Assess_Detected_Object  FROM Swarm_Operator
    DO ADD $a PRECEDES $b; OD;

COORDINATE  $a: Issue_Recovery_Command      FROM Swarm_Operator
     DO  COORDINATE <!> $b: End_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!>  $a: Present_Object_Signature    FROM Environment,
        $b: Detect_Object               FROM Swarm
        DO ADD $a PRECEDES $b; OD;
```

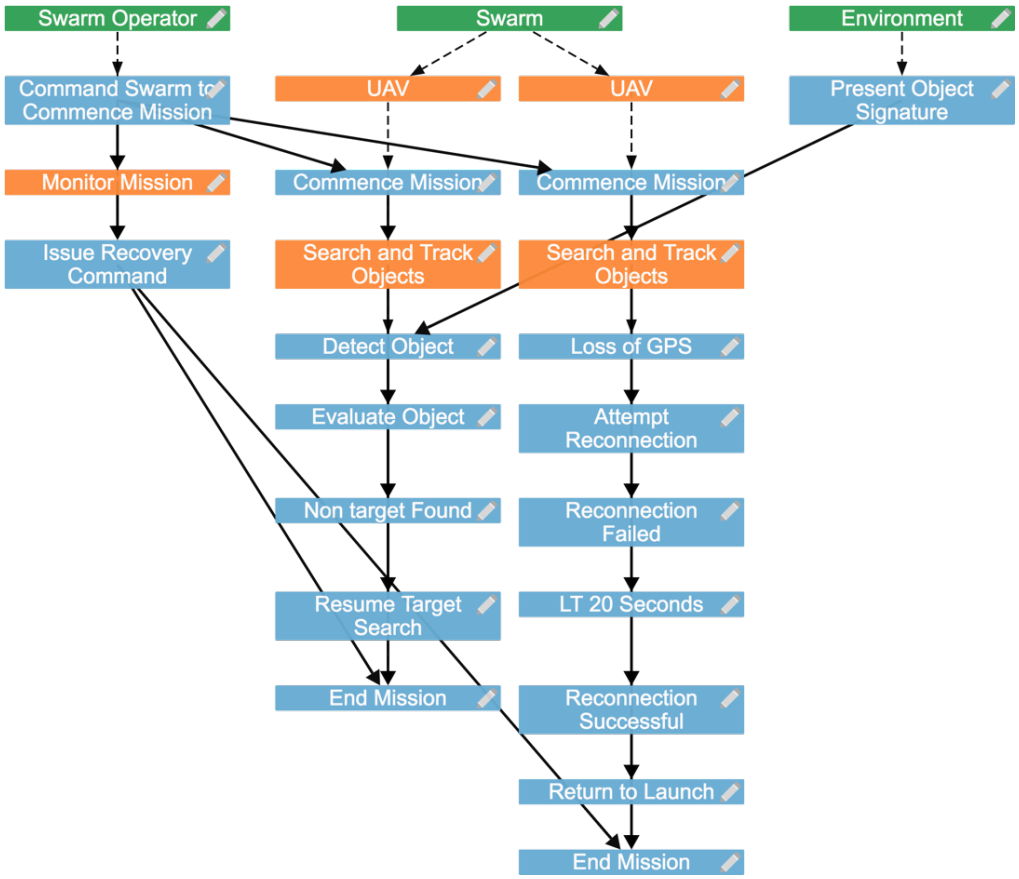
LT = less than and GT = greater than

Figure 27. MP Code – Loss of GPS



LT = less than and GT = greater than

Figure 28. MP Sequence Diagram – Loss of GPS



LT = less than and GT = greater than

Figure 29. MP Sequence Diagram – Loss of GPS (Scope = 2)

6. Loss of Link to Payload

Unmanned aerial vehicles serve many purposes and the payloads they employ make those purposes possible. Specifically for SAR missions, imaging payloads are considered integral to the success of the mission. To mitigate payload failures, there are many design factors to consider, which include (Johnson, 2012):

- available mass, volume and power budgets
- video interfaces and available wires
- bandwidth
- mission duration
- slant range or distance from UAV camera to target
- on-board image processing

> To complicate matters, consider what is needed to deliver narrow-field-of-view, quality video from a small aircraft. The platform is moving, vibrating, and subject to random and uncontrollable motion of the aircraft. The line of sight for the delivered video needs to point at the object of interest on the ground, hold on that object of interest while the aircraft is flying its course, and be immune to input disturbances that would cause the camera's line of sight to move. (Johnson, 2012).

The overwhelming need for failsafe behaviors among payloads quickly becomes apparent.

Failsafe Behavior: "Electronics failures are reported for about 25% of all failures, the rest being attributed to weather and pilot error" (Caswell and Dodd, 2014). Therefore, the first failsafe behavior to activate during a payload loss of link failure is to cycle or reboot the payload controller. If a reboot does not correct the issue, the next step in the process is to bring the UAV home, or RTL, for troubleshooting. If the payload failure has resulted in a GCS loss of link, the UAV should autonomously land.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object
      Evaluate_Object
        ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Payload_Loss_of_Link
    /*Failsafe Behavior*/
    Controller_Reboot
    (
      ( Failure_Remains
        ( GCS_Loss_of_Link Auto_Land |
          Take_Manual_Control Return_to_Launch
        ) |
      Failure_Mitigated Resume_Target_Search
      )
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
    DO COORDINATE <!> $b: Commence_Mission    FROM Swarm
        DO ADD $a PRECEDES $b; OD;
    OD;

COORDINATE <!> $a: Track_Target      FROM Swarm,
    $b: Assess_Detected_Object  FROM Swarm_Operator
        DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command    FROM Swarm_Operator
    DO COORDINATE <!> $b: End_Mission    FROM Swarm
        DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
    $b: Detect_Object                   FROM Swarm
    DO ADD $a PRECEDES $b; OD;
```

Figure 30. MP Code – Loss of Link to Payload

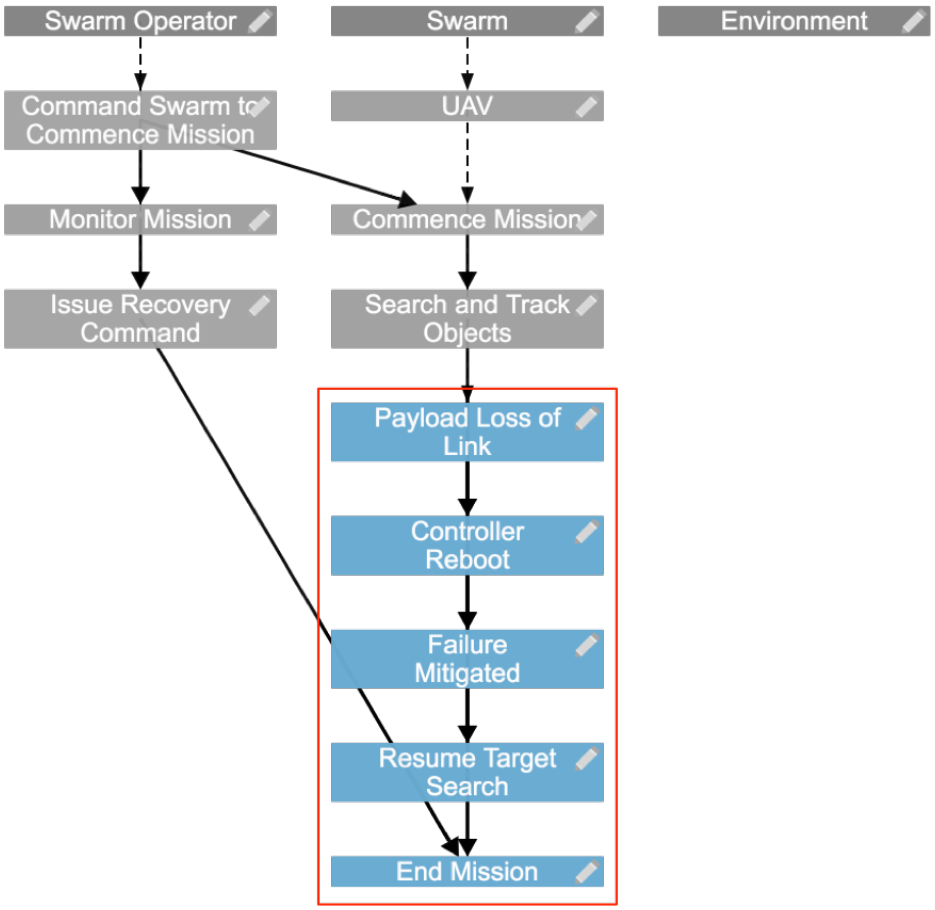


Figure 31. MP Sequence Diagram – Loss of Link to Payload

7. Geofence Breached

Whether manned or unmanned, most SAR missions are constrained by a search area perimeter. Through algorithmic perimeter planning, a search area is defined based on the highest probability of success. This area can expand exponentially as each hour passes. In UAV terminology, this perimeter is known as a geofence, or a virtual barrier that constrains the air vehicle to a certain area. With unmanned swarm missions, a geofence is usually defined by highest area for probability of rescue or by command and control communication limitations. A geofence breach occurs when one of the UAVs exits the predefined geofence boundary. A breach can have serious safety implications and must be handled immediately. Furthermore, a breach resulting in the loss of communications is difficult to reverse. To avoid complete communications blackout, the

area defined by the geofence may be reduced to provide a communications buffer just outside of the geofence.

Failsafe Behavior: When a UAV ignores a directive to remain within a defined area, it is assumed that something rather significant is occurring within the UAV's autopilot control. Therefore, the default failsafe behavior for a geofence breach is to take over manual control of the air vehicle and return it to the launch location. If manual override is unsuccessful, the follow up failsafe behavior is to kill the throttle and enter into a controlled crash.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  ( Detect_Object
    Evaluate_Object
      ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Geofence_Breached
    /*Failsafe Behavior*/
    Take_Manual_Control
    (
      Manual_Control_Failed Kill_Throttle Controlled_Crash |
      Return_to_Launch
    )

  )
*);

COORDINATE  $a: Command_Swarm_to_Commence_Mission   FROM Swarm_Operator
     DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD; /* 10b */

COORDINATE  <!> $a: Track_Target    FROM Swarm,
        $b: Assess_Detected_Object  FROM Swarm_Operator
    DO ADD $a PRECEDES $b; OD;

COORDINATE  $a: Issue_Recovery_Command      FROM Swarm_Operator
     DO  COORDINATE <!> $b: End_Mission     FROM Swarm
           DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!>  $a: Present_Object_Signature    FROM Environment,
        $b: Detect_Object               FROM Swarm
        DO ADD $a PRECEDES $b; OD;
```

Figure 32. MP Code – Geofence Breach

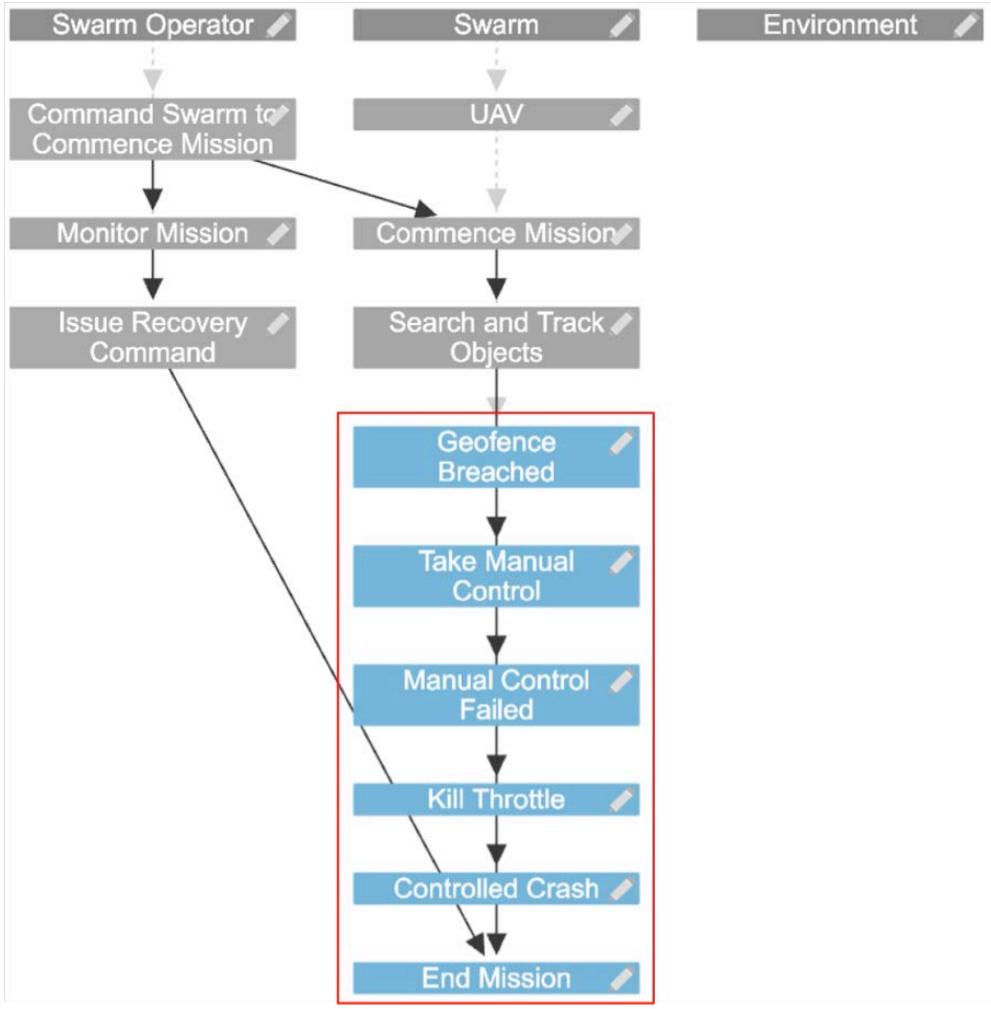


Figure 33. MP Sequence Diagram – Geofence Breach

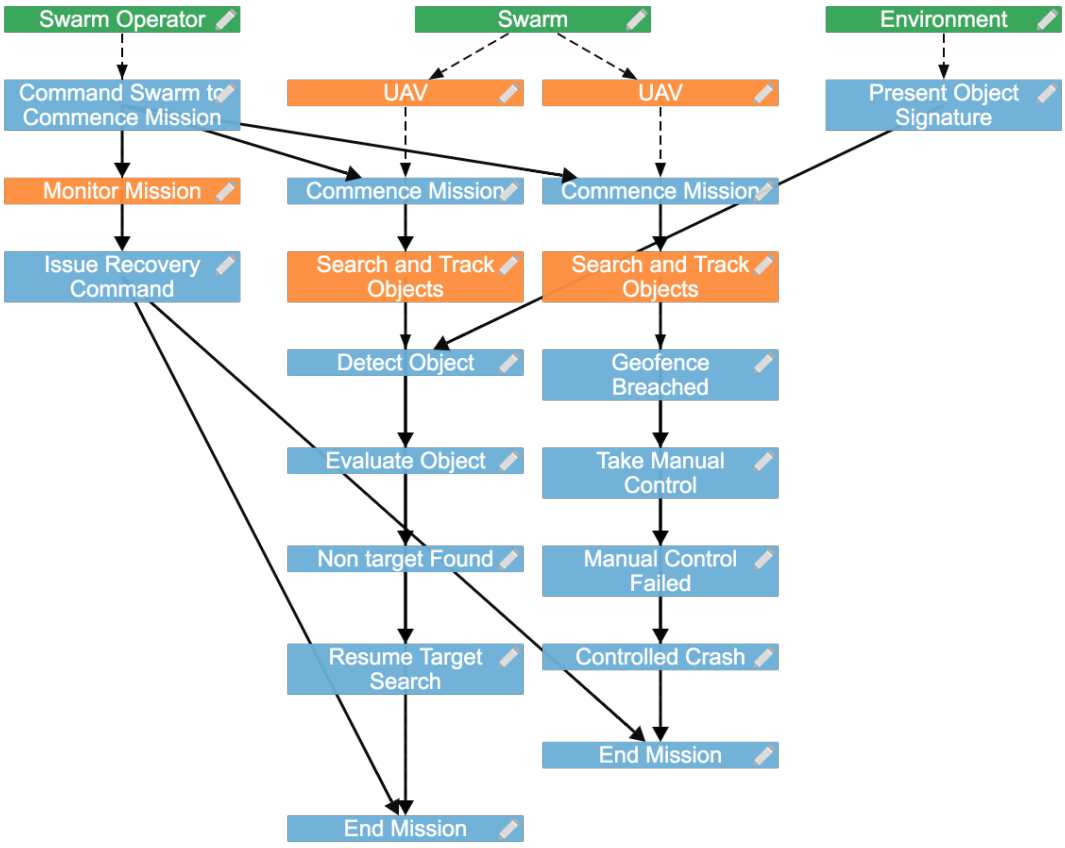


Figure 34. MP Sequence Diagram – Geofence Breach (Scope = 2)

B. FAILURE MODE PRIORITIZATION

The previous section described a detailed analysis of potential failure modes and failsafe behaviors through building isolated MP models. When planning for potential failure modes and corresponding failsafe behaviors, it is important to place priorities or weights to each failure mode. Whether the failure mode monitoring software is active or passive, most UAV health monitoring systems use an order of operations or flow chart approach to determine the best mitigation strategy for a given failure mode. These automatic systems can be developed and integrated onboard a UAV to reduce its vulnerability to errors and failures in the actuators, control surfaces and sensors (Rabbath, Alain, and Léchevin 2010). As detailed below, there are multiple reasons for this requirement.

- Catastrophic failure modes: There are certain disastrous failure modes that, upon occurrence, must be dealt with immediately and all other failure modes become trivial.
- Simultaneously occurring failure modes: It is not uncommon for failure modes to emerge concurrently, forcing a decision on which failure mode should be treated first. This is one of the key benefits to using an order of operations approach for health monitoring. As seen Figure 35, there is no situation where two or more concurrent failure modes can break the model.
- Shared failsafe behaviors: In taking a more object-oriented approach, there are several failure modes that have the same failsafe behavior (e.g., Return to Launch). An order of operations allows for the declaration of one failsafe method for multiple failure modes.

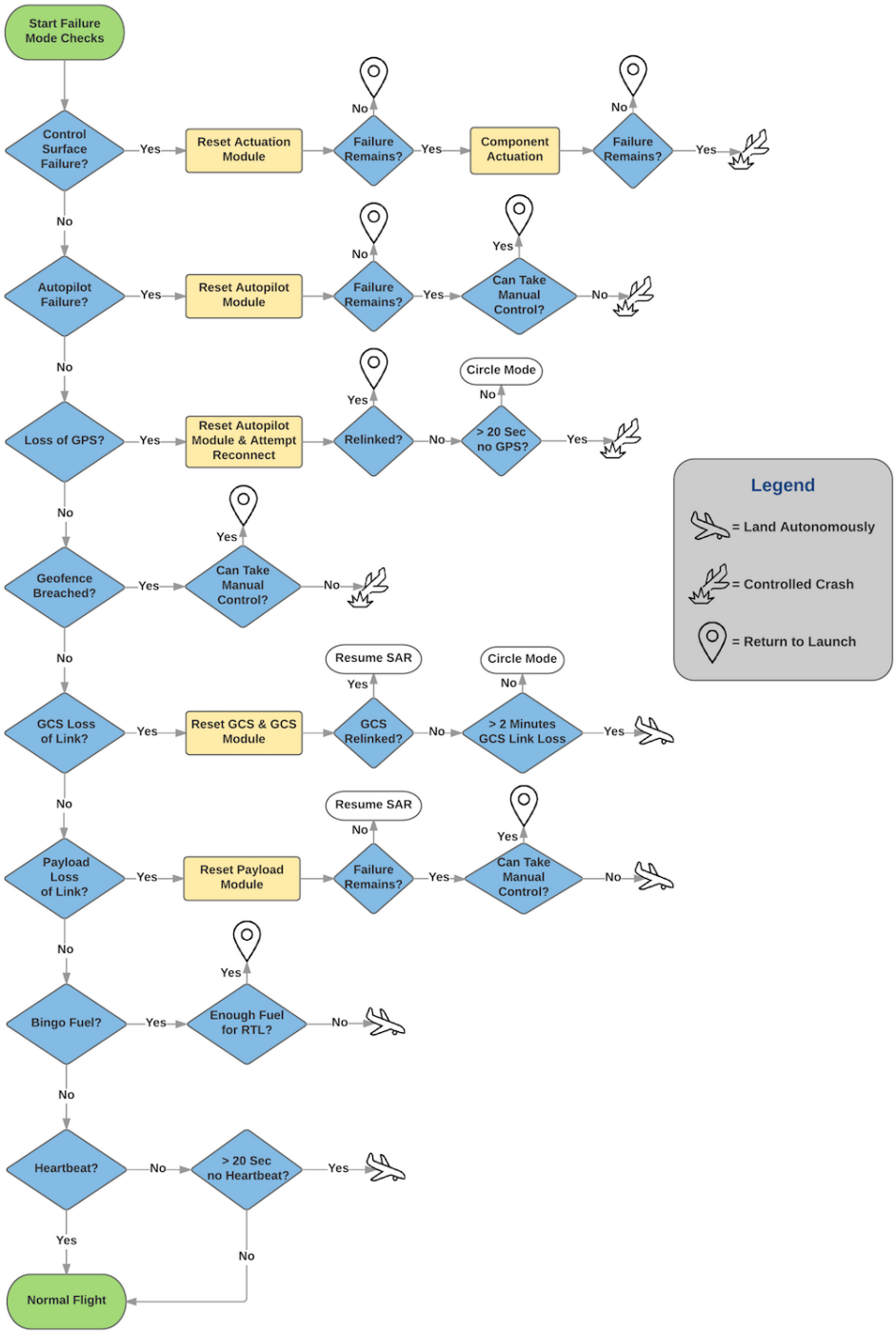


Figure 35. Failure Mode and Failsafe Behavior Order of Operations

To expand further, MP can be used in an object-oriented prioritization approach to evaluate the impact of numerous potential failure modes. After an exhaustive list of

potential failure modes and failsafe behaviors has been identified, they should be analyzed and prioritized. Through numerous modeling tests, prioritization tends to follow a more drastic to less drastic failsafe behavior pattern. In Table 3, all failure modes ranked near the top share the failsafe behavior of controlled crash. Once priorities have been assigned, each behavior should be reviewed for commonality. Specifically, failsafe behaviors have a strong possibility of replication, which allows the modeler to generate a single method to address multiple failure mode scenarios. For example, both a geofence breach and loss of GPS share the failsafe behavior to enter a controlled crash.

Table 3. Failure Mode Priority and Commonality

Rank	Failure Mode	Failsafe Behavior	Controlled Crash	Auto Land	RTL
1	Control Surface	Controller Reboot - Failure Remains? Yes: Component Actuation - Failure Remains? Yes: Controlled Crash No: Return to Launch No: Return to Launch	X		X
2	Autopilot	Attempt to take Manual Control - Manual Control > Return to Launch - No Manual Control > Controlled Crash	X		X
3	Loss of GPS	Attempt Reconnect - Reconnected > Return to Launch - Not Reconnected > Longer than 20 seconds? Yes: Controlled Crash	X		X
4	Geofence Breached	Take Manual Control - Manual Control Failed > Controlled Crash - Manual Control Success > Return to Launch	X		X
5	GCS Link Loss	Attempt to take Manual Control - Manual Control > Return to Launch - No Manual Control > Auto Land		X	X
6	Payload Link Loss	Controller Reboot - Failure Remains? Yes: GCS Linked? No: Auto Land Yes: Return to Launch		X	X

Rank	Failure Mode	Failsafe Behavior	Controlled Crash	Auto Land	RTL
		No: Resume Target Search			
7	Bingo Fuel	Check Fuel Level - Enough to Return? > Return to Launch - Not Enough to Return? > Auto Land		X	X

The following MP model combines the failures modes defined in Table 3 through a prioritized and object-oriented methodology. On the UAV, these checks would occur in a feedback loop with a heartbeat-like return of the UAV's status. The status would most likely indicate a normal operation or presence of failure. Many of the failures have automatically engaged failsafe behaviors, while others require the intervention of a human operator to make a decision on how to proceed. To avoid the chance of ignoring a high priority failure mode, the loop continues to check for failure modes, despite the state of the current failure. This allows for the interruption and reassignment of tasking.

```
    /******Failure Modes******/

    /*Control Surface Failure*/
    Control_Surface_Failure Reset_Actuation_Module
    (
      ( Failure_Remains Component_Actuation
        ( Failure_Remains Controlled_Crash |
          Failure_Mitigated Return_to_Launch
        )
      ) | Failure_Mitigated Return_to_Launch
    ) |

    /*Autopilot Failure*/
    Autopilot_Failure Reset_Autopilot_Module
    ( Failure_Remains Attempt_Manual_Control
      ( Manually_Controlled Return_to_Launch |
        Not_Manually_Controlled Controlled_Crash
      ) |
      Autopilot_Restored Return_to_Launch
    ) |

    /*Loss of GPS Failure*/
    Loss_of_GPS Reset_GPS_Module Attempt_Reconnect
    ( Reconnected Return_to_Launch | Not_Reconnected
      ( LT_20_Seconds Wait | GT_20_Seconds Controlled_Crash )
    ) |

    /*Geofence Breached Failure*/
    Geofence_Breached Attempt_Manual_Control
    ( Manually_Controlled Return_to_Launch |
      Not_Manually_Controlled Controlled_Crash
    ) |

    /*GCS Loss of Link Failure*/
    GCS_Loss_of_Link Reset_GCS Reset_GCS_Module Attempt_Reconnect
    ( Reconnected Resume_Target_Search |
      Not_Reconnected ( LT_2_Minutes Wait |
                        GT_2_Minutes Auto_Land )
    ) |

    /*Payload Loss of Link Failure*/
    Payload_Loss_of_Link Reset_Payload_Module
    ( Failure_Remains Attempt_Manual_Control
      ( Manually_Controlled Return_to_Launch |
        Not_Manually_Controlled Auto_Land
      ) | Failure_Mitigated Resume_Target_Search
    ) |

    /*Bingo Fuel Failure*/
    Bingo_Fuel Check_Fuel_Level
    ( Adequate_Fuel Return_to_Launch |
      Inadequate_Fuel Auto_Land )
    )
*);

Auto_Land:
  Track_Current_Position
  Reduce_Altitude
  Enter_Reverse_Helix
  Land
  Power_Down;

Controlled_Crash:
  Track_Current_Position
  Kill_Throttle;

Return_to_Launch:
  Clearout_Existing_Waypoints
  Set_Final_Waypoint
  Travel_to_Final_Waypoint;
```

Figure 36. MP Code – Combined Failure Modes

C. PATTERNS

The list of potential failure modes can grow much larger than the seven descried in the previous section. Through the iterations of failure mode insertions and analyses, a common pattern emerged, and once recognized, made it easier to understand the impact a

failure mode can have on an a set of event traces. In MP, a failure is presented as an alternative event that can occur based on where the failure mode is placed within the model. If a failure is place at the root level for a sequence of events, the failure will override the entire operation, as with many of the SAR mission failure modes presented in this thesis.

```
SCHEMA Patterns_for_Failure_Mode_Insertion

ROOT System:
(
  ( Do_Activity_1
    ( Do_Activity_2
      | Fail_to_do_Activity_2 )
    | Fail_to_do_Activity_1 )
  | Fail_to_do_any_Activity /* Failure at Root Level */
);

/* Failsafe Behaviors */
Fail_to_do_Activity_1: Do_failsafe_1;
Fail_to_do_Activity_2: Do_failsafe_2;
Fail_to_do_any_Activity: (Do_failsafe_3a | Do_failsafe_3b);
```

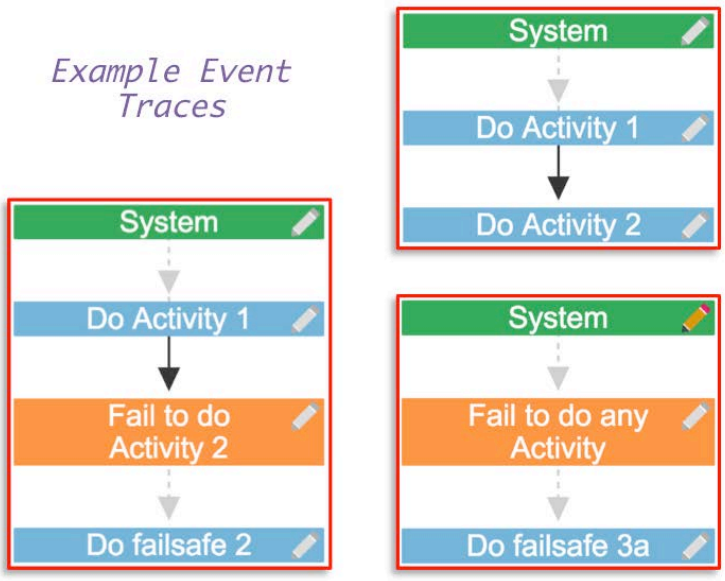


Figure 37. MP – Failure Mode and Failsafe Behavior Pattern

These patterns for failure mode insertion provide a basis for understanding and planning for failure mode analysis in MP. This pattern is used in several of the failure modes presented in the previous section, but is not a one-size-fits-all approach. Though

discovering patterns and formulas make complex tasks easier, it is import to remember that as patterns are realized, complacency and assumptions are avoided. This mode of thinking is similar to the patterns a medical doctor follows every time a diagnosis is made. If steps are skipped or forgotten about due to complacency, the consequences can be devastating.

V. ANALYSIS OF FINDINGS AND RESULTS

A. KEY FINDINGS

1. Event Trace Analysis

When the behaviors between actors are effectively modeled in MP, the analytical benefits are realized as the modeler steps through each event trace. Through this ability to view every possible interaction, anomalies and abnormalities in the modeled behaviors can be revealed. There are additional capabilities offered by MP that make this process even easier.

Assertion Checking: As with any modeling approach, the models and underlying programing needs to be debugged. Given the exhaustive set of event traces produced by MP, this can be a rather overwhelming. Monterey Phoenix offers the CHECK construct that was used in this research to automate the event trace monitoring. This assertion checking approach marks any traces that violate an assigned Boolean expression (Auguston 2016). Once the model was proven to be efficient and bug free these CHECKs were removed.

Trace annotation: For this research the CHECK clause provided the level of detail necessary to adequately debugging the SAR models. In more complex models, MP also offers trace annotations to make messages more specific and focused. Instead of a generic message about what is occurring, variable-based values from the model are also captured in the message.

```
CHECK       #Assess_Detected_Object FROM Swarm_Operator ==
            #Object_is_Valid_Target FROM Swarm_Operator
ONFAIL SAY(“Object is not valid”);
```

2. Concurrent Failure Mode Modeling

As mentioned in the section on “Failure Mode Prioritization,” it is beneficial to prioritize and continuously check for failure modes in a loop. When considering the programming logic that would be required to create this heartbeat like feedback loop, the

most efficient approach is to bundle these potential failure scenarios into a single module or class.

Through studying the benefits of using MP to inspect the impact of failure modes on a swarm, it was ultimately determined that MP is much more powerful when analyzing each failure mode in a phased approach, where each failure mode is modeled in an isolated manner. When every failure mode and failsafe behavior is modeled concurrently, the ability to inspect the impacts becomes extremely difficult. Furthermore, it becomes a practice in futility as the scope is increased on a simultaneous analytical modeling approach.

3. Failsafe Behavior Consistency and Commonality Validation

As revealed during this research, failure modes may require mitigation through common failsafe methods. This is not necessarily a discovery, but more of a hypothesis or assumption that the majority of failure modes are distinctive in their behaviors, but often, assumptions cloud the researchers ability to recognize emergent patterns between failsafe behaviors.

Once the Monterey Phoenix recipe for modeling failure modes and failsafe behaviors was understood, a systematic and methodical approach was taken to model each failure mode in an isolated environment against the baseline SAR mission. This approach was an iterative process contained within each failure mode analysis period. Once it was determined that a failure mode had reached its optimal modeling solution, the next failure mode analysis period began.

After modeling all failure modes, the process of failure mode prioritization started. As discussed in the prioritization section, setting an order of precedence is necessary to properly handle the more significant or detrimental failure modes first. Given the incremental approach to modeling individual failures, it was not until this point that inconsistencies among similar failsafe behaviors started emerging.

An example of a situation where an inconsistency was identified can be seen in the analysis of the failure modes, “payload loss of link” and “geofence breach.” During

the examination each, the failsafe behavior “return to launch” (RTL) was identified. While this is a logical behavior for both scenarios, the geofence breach failure is possibly tied to erroneous waypoints and the RTL failsafe behavior included clearing out all waypoint before returning to launch. The payload loss of link failure also had a failsafe behavior of RTL, but while the clearing of waypoints is also applicable, it was not included. It was not until these failures were modeled together that a common RTL method was identified that applied to all RTL behaviors.

From this lessons learned activity, a new “failsafe behavior consistency validation” method to planning for failure modes was developed. In contrast to analyzing each failure mode and corresponding failsafe behavior in a vacuum, failure modes modeled and examined together in a staged approach appear to provide greater benefit. In this research, for example, the failure mode identification, failure mode model placement, failure mode impact analysis, and failsafe behavior employment could be used as consistency and commonality validation stages. Instead of iterating through each failure mode and failsafe behavior individually, all failure modes would work through each stage concurrently. A new hypothesis for this phased approach could be written as the following:

> Failure mode and failsafe behavior consistencies are simplified if each step to analyze individual failure modes is staged and all failure modes iterate through each stage concurrently.

Employment of the above hypothesis predicts that common failsafe behaviors will be identified earlier in the analysis process, thus allowing for more rapid model optimization. Recognizing common failsafe behaviors across all failure modes creates a shared method for each failure mode to call that provides an object-oriented approach for failure modes to leverage. Additionally, as new failure modes are discovered, the recreation of failsafe behaviors may become unnecessary.

Another example of this approach is demonstrated in Table 3: Failure Mode Priority and Commonality. The failsafe behaviors, auto land, controlled crash, and return to launch became methods for all other failure modes to call if the behavior was applicable.

B. SURPRISING AND UNEXPECTED OUTCOMES

1. Bingo Fuel Emergent Failsafe Behavior

As seen in Figure 38, during initial examinations of the "bingo fuel" failure mode, the obvious failsafe behavior was to return home or auto land. After close examination of the event traces generated from the model (see Figure 39), there were instances where the UAV would return to launch right after finding a potential target. This was not necessarily a modeling error but more of an identification of emergent or better-suited behavior.

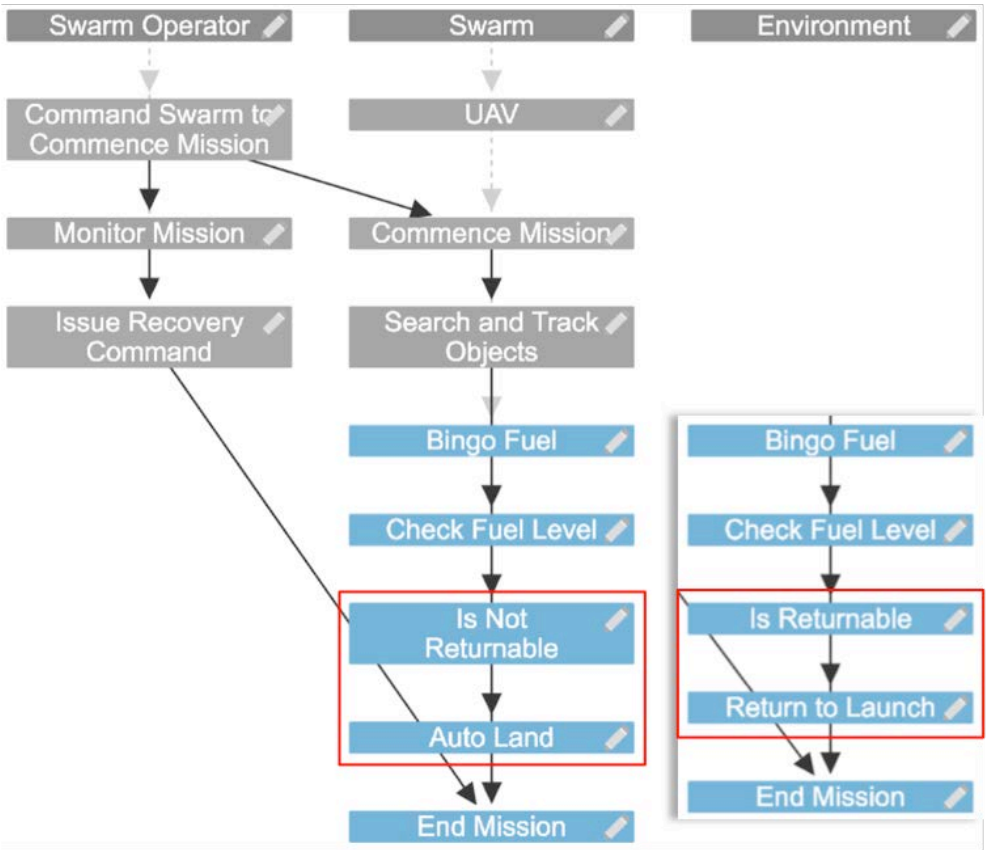


Figure 38. MP Sequence Diagram – Bingo Fuel (Initial Analysis)

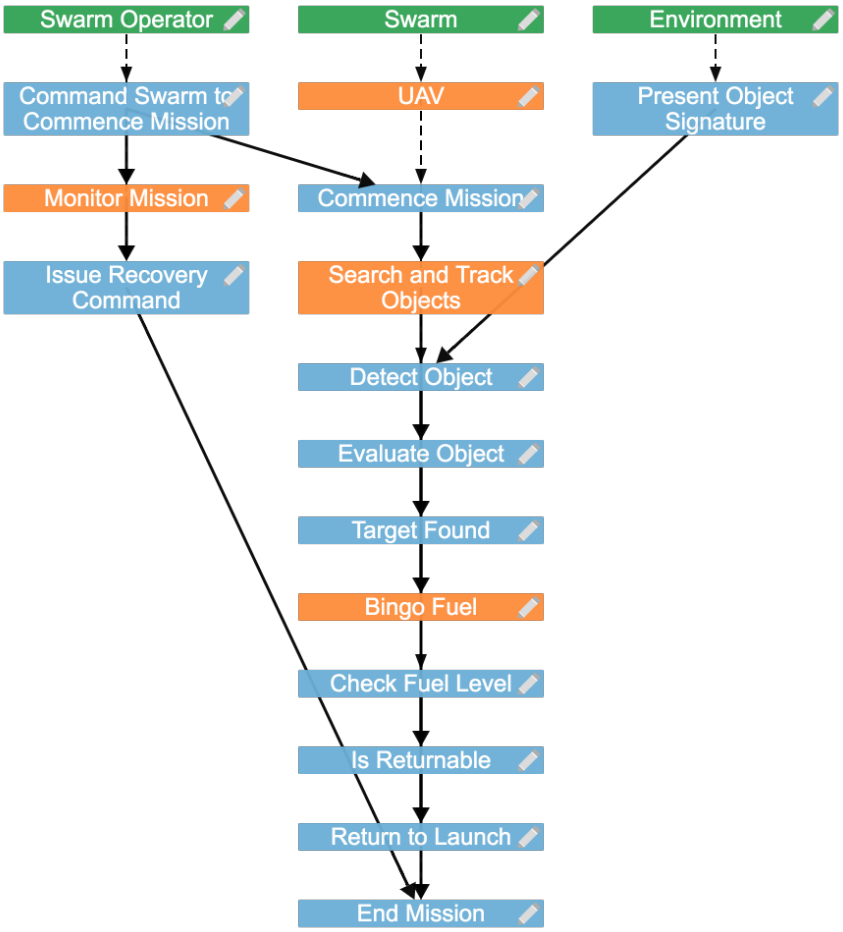


Figure 39. MP Sequence Diagram – Bingo Fuel (Initial Analysis)

Failsafe behaviors are often necessary deviations from behaviors considered detrimental to the mission. Many may conclude that a failsafe behavior that is not triggered correctly consequentially becomes a failure mode. This may hold true in most situations where the value of the asset, or UAV in this research, is more valuable than the mission. When the value of the mission is greater than the assets involved, there lies a great opportunity for the emergence of new behaviors.

When unmanned swarms of UAVs are performing SAR missions for lost humans (e.g., at sea), the value of the mission is most definitely more valuable than the asset. In this situation, it's important to reevaluate the failure modes and consider how the mission may better be served if adjustments were made to the failsafe behaviors. Again, in the

case of the bingo fuel failure mode, once this reexamination occurred, two emergent behaviors were realized.

Sacrifice the UAV – As discussed in the failure modes section, when a bingo fuel event occurs, there is just enough fuel to return to launch or some predetermined location. In this scenario, when the UAV enters bingo fuel, the new failsafe behavior is to reassess the SAR environment before returning to launch. If during that reassessment a target or person in distress (PID) is identified, the swarm operator has the option to allow the UAV to continue tracking the target. Of course, this will ultimately result in the expiration of onboard fuel and loss of aircraft. That result pales in comparison to the benefits gained from added tracking details on a lost target, which may result in lives saved. Figures 40 and 41 illustrate the new MP model and sample event trace containing the new failsafe decision.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |
    /* Failure Mode */
    Bingo_Fuel
    /* Failsafe Behavior */
    Reassess_Environment
    (
      ( Target_Found Track_Target) |
      ( Non_target_Found Check_Fuel_Level
        (
          Is_Returnable Return_to_Launch |
          Is_Not_Returnable Auto_Land
        )
      )
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
    DO COORDINATE <!> $b: Commence_Mission    FROM Swarm
         DO ADD $a PRECEDES $b; OD;
    OD;

COORDINATE <!> $a: Track_Target      FROM Swarm,
    $b: Assess_Detected_Object  FROM Swarm_Operator
         DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command    FROM Swarm_Operator
    DO COORDINATE <!> $b: End_Mission    FROM Swarm
         DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
    $b: Detect_Object                       FROM Swarm
    DO ADD $a PRECEDES $b; OD;
```

Figure 40. MP Code – Bingo Fuel (UAV Sacrifice)

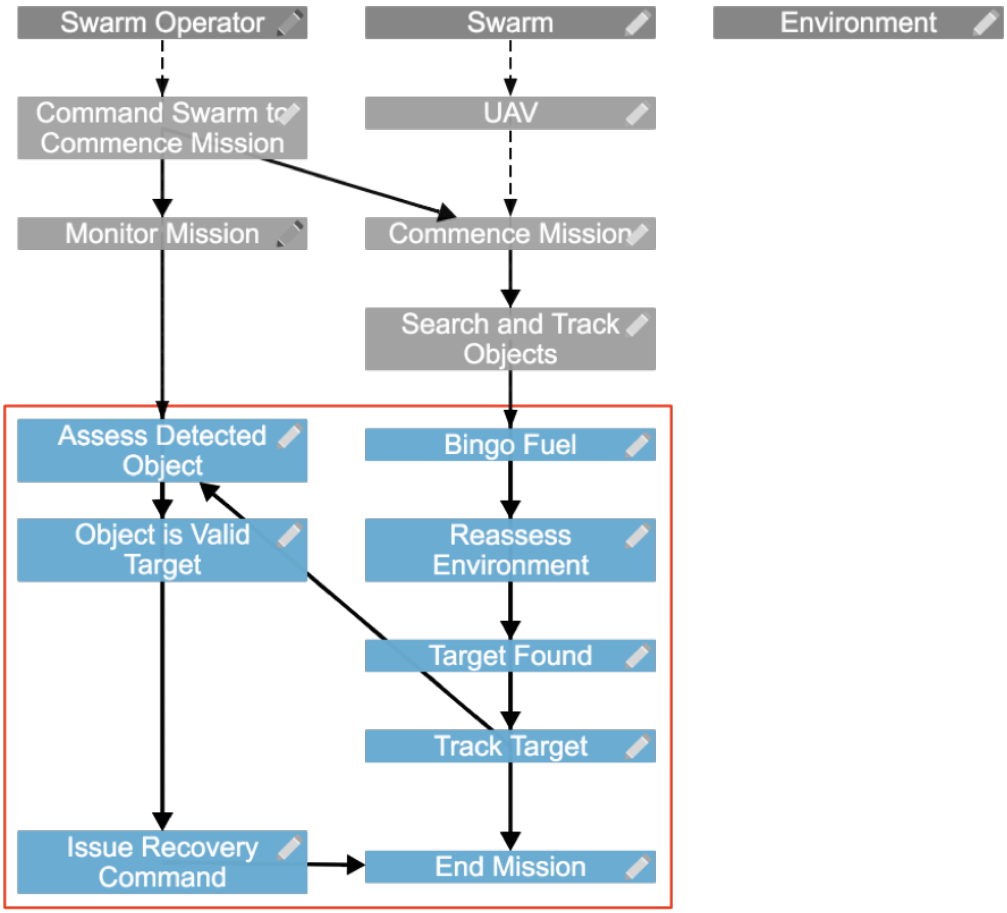


Figure 41. MP Sequence Diagram – Bingo Fuel (UAV Sacrifice)

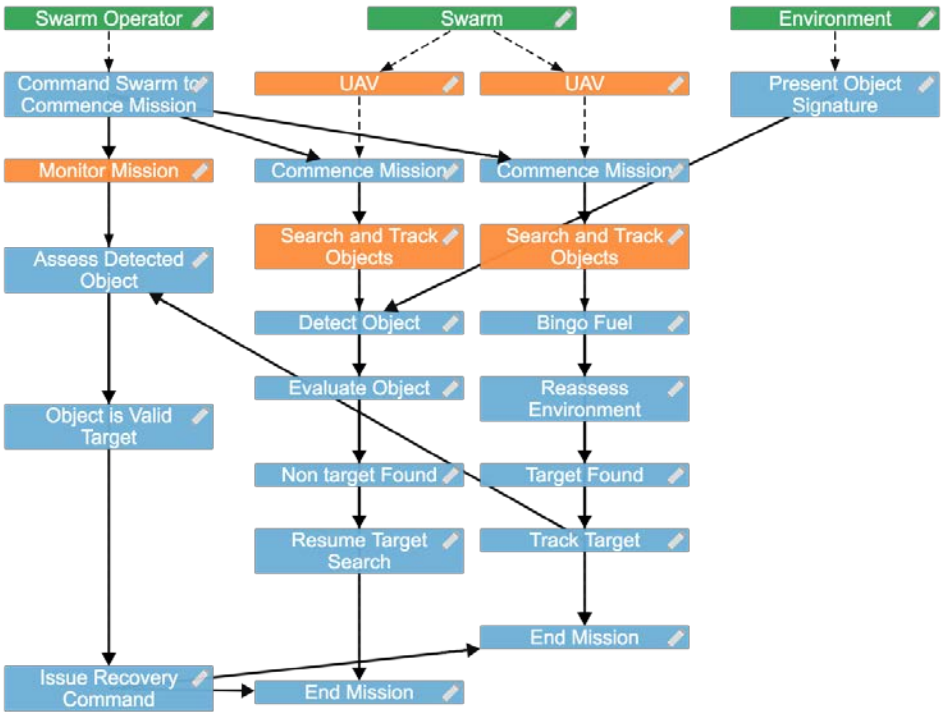


Figure 42. MP Sequence Diagram – Bingo Fuel (UAV Sacrifice, Scope = 2)

UAV Relief – As more thought and analysis was performed on the bingo fuel failure modes and the new failsafe decision to sacrifice the UAV, another failsafe decision option emerged. Depending on the size of the swarm, the swarm operator may want to have the option to recover the bingo fueled UAV and allow for the swarm to compensate for the UAV that has returned home. If employed in theatre, this option would require a great deal of logistics and complicated algorithms to calculate the optimal swarm coverage without allowing the swarm to be spread too thin.

This approach involves a new operation to allow the nearest UAV to relieve the bingo fueled UAV. Once the new UAV arrives, the UAV experiencing the bingo fuel failure mode checks its fuel level and decides to return to launch or auto land. In addition to the complicated algorithms required to coordinate this activity, the formula to calculate the bingo fuel would need to be adjusted to include the distance of the nearest UAV. This would be a constantly changing value as the distances are continuously changes.

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target | Non_target_Found Resume_Target_Search ) |
    /* Failure Mode */
    Bingo_Fuel
    /* Failsafe Behavior */
    Request_Relief
    Nearest_UAV_Arrives
    Check_Fuel_Level
    (
      Is_Returnable Return_to_Launch |
      Is_Not_Returnable Auto_Land
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
    DO COORDINATE <!> $b: Commence_Mission    FROM Swarm
        DO ADD $a PRECEDES $b; OD;
    OD;

COORDINATE <!> $a: Track_Target      FROM Swarm,
    $b: Assess_Detected_Object  FROM Swarm_Operator
        DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command    FROM Swarm_Operator
    DO COORDINATE <!> $b: End_Mission    FROM Swarm
        DO ADD $a PRECEDES $b; OD;
    OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
    $b: Detect_Object                  FROM Swarm
    DO ADD $a PRECEDES $b; OD;
```

Figure 43. MP Code – Bingo Fuel (UAV Relief)

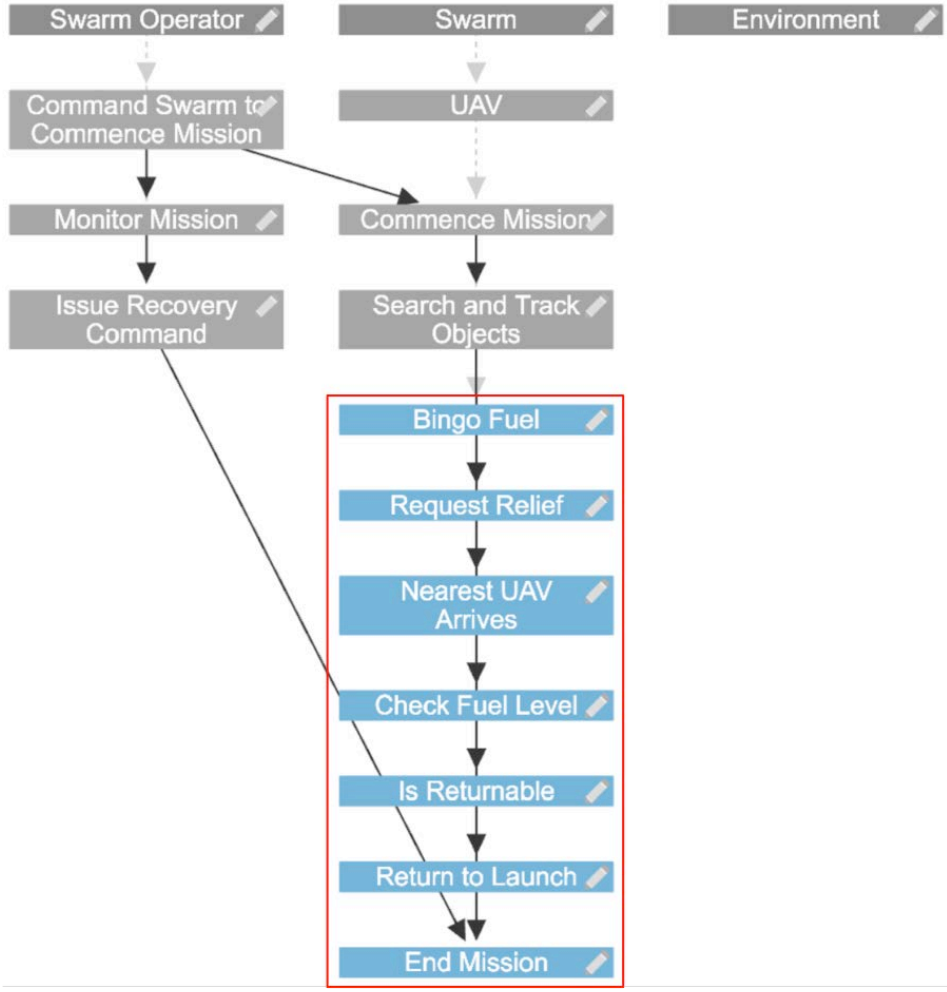


Figure 44. MP Sequence Diagram – Bingo Fuel (UAV Relief)

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

A. BENEFITS OF STUDY

Understanding the impacts of failure modes and how they can influence a mission can provide great benefit to mission planning. This not only applies to the swarming search and rescue (SAR) missions presented in this research, but to any mission where advanced planning is required. Modeling the CONOPS of the SAR mission in Monterey Phoenix (MP) proved to provide valuable insight into identifying failure modes and failsafe behaviors.

A product of this research included a small catalog of prioritized failure modes and corresponding failsafe behavior for fixed wing UAVs. The failure modes modeled in this thesis are closely related to the SAR mission, but have applicability in various other single UAV and swarming UAV missions.

Through this research of modeling failure modes and failsafe behaviors, a common pattern emerged. This pattern is not design or mission specific and can be used to model and understand how failure impact a system. By leveraging a straightforward and repeatable "do activity → activity failed → do failsafe" pattern, the modeler is able to rapidly insert mitigation behaviors throughout the model for unwanted failure modes.

An interesting and beneficial discovery revealed through this research was the discovery of several emergent behaviors. While modeling each failure mode and comparing event traces for each failsafe behaviors, situations and scenarios that would not have otherwise been revealed, or at least not as quickly, emerged. The bingo fuel scenario was one of the most notable emergent behaviors. The default assumed behavior was to return to launch when fuel was running low, but careful examination of the event traces revealed the importance of comparing mission importance to asset importance. With the SAR mission, it is obviously more important to locate and potentially save a person or persons in distress (PID). This shift of importance from asset to mission revealed an alternate behavior to have the bingo fueled UAV reassess the environment for PIDs. If a PID is discovered, the UAV will now stay out longer to gather geospatial

data about the PID's location. This will potentially result in the loss of the UAV, which is trivial when considering the possibility of saving a person's life.

B. ITEMS FOR FURTHER STUDY

Monterey Phoenix has many uses outside what was revealed in this thesis. While working through this study, several areas of opportunity for furthering the research were identified. The following section discusses these potential follow-on opportunities.

1. Bingo Fuel Expansion

A surprising discovery with the bingo fuel scenario was discovered late in this thesis research that provides two opportunities to expand the research into MP's capabilities. As discussed, MP facilitated the ability to recognize emergent behaviors through weighing importance of the mission to the asset in the mission.

One of those behaviors was to leverage neighboring UAVs in the swarm to provide relief to the bingo fueled UAV. This approach would provide continuous coverage for the search area, as the other UAVs spread out. This approach would require a significant amount of computational analysis to efficiently manage the swarm. There are two main follow-on opportunities with this method.

A substantial optimization problem exists for calculating the area of coverage for each UAV and how that area would change based on the relief situation. Minimizing the values for the optimal coverage overlap areas would be a worthy objective function. Additionally, considerations or constraints would need to be made for the time required for a neighboring UAV to arrive. This value would need to be wrapped into the bingo fuel calculation to provide ample time for response.

A second opportunity lies in expanding the MP model. As seen in Figure 45, only the failure and failsafe behaviors for a single UAV are modeled. The model does not account for which UAV should provide the relief. There exists an opportunity to update the model to establish relationships between each UAV in the swarm. These relationships could include the reactionary behavior of the UAV providing relief as well as the swarm operator.

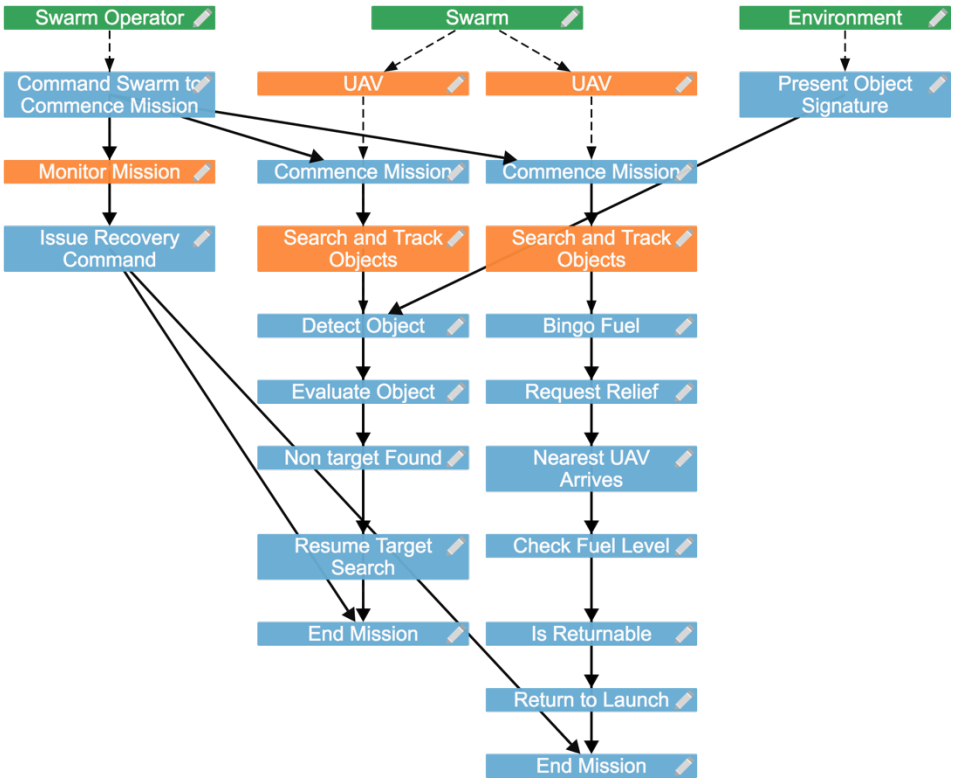


Figure 45. MP Sequence Diagram – Bingo Fuel (UAV Relief, Scope = 2)

2. Inter-UAV Communications

Monterey Phoenix offers the ability to model the communication transmissions between actors. An additional area for potential future study is to use MP to model the inter-UAV communications among swarms through a publish-subscribe pattern. The current version of firebird.nps.edu contains example 11, a publish-subscribe model that could be used as a foundation (Auguston and Giammarco 2016).

3. Swarm vs. Swarm Model Expansion

The Department of Defense (DOD) is starting place a fair amount of attention on the air-to-air combat abilities of swarming unmanned aerial systems (UAS) and more specifically, how swarms might behave when fighting against other swarms. As briefly discussed in the small scope hypothesis section, MP has the ability to provide valuable insight into a swarm vs. swarm. A future research item might be to model the behaviors

within each swarm group and the interactions between adversary swarms to determine the most optimal way to engage in aerial combat.

4. Monterey Phoenix Phased Modeling Approach

As revealed in the section on failure mode prioritization, the ability to model multiple failure modes in one schema is beneficial for identifying commonalities, but imposes a great time cost on MP's event trace performance. A valuable area of research would be to address these large models with multiple decision paths. This was briefly discussed in Chapter III of this thesis, but one recommendation is a phased approach, where areas of the mission would be extracted and assigned to an individual phase (e.g., pre-launch, takeoff, ingress, egress and landing) and modeled discretely. It is important to watch for behaviors that cross over or effect more than one phase, as that will need to be accounted for in the model.

5. Cross-Domain Pattern Recognition

A useful outcome of this thesis was the development of a pattern for modeling failure modes and failsafe behaviors. Patterns can be extremely valuable in developing expedient behaviors. A follow-on research effort could expand on this principle by identifying behavior patterns that repeat across different MP architecture models. Once identified, a playbook of sorts with these reusable patterns could provide benefit to the MP user community.

APPENDIX: MP CODE

The following section provides a complete plaintext searchable list of all Monterey Phoenix Code used for the purposes of this thesis.

Swarm SAR Mission - Baseline

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Control Surface Failure

```
SCHEMA Swarm_Search_and_Track
```

```
ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Control_Surface_Failure
    /*Failsafe Behavior*/
    Controller_Reboot
    (
      ( Failure_Remains
        Component_Actuation
        ( Failure_Remains Attempt_Controlled_Crash |
          Failure_Mitigated Return_to_Launch
        )
      ) |
      Failure_Mitigated Return_to_Launch
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Autopilot Failure

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
```

```
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Autopilot_Failure
    /*Failsafe Behavior*/
    Attempt_Manual_Control
    (
      Manual_Control_Success Return_to_Launch |
      Manual_Control_Fail Controlled_Crash
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Loss of GPS

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);
```

```
ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Loss_of_GPS
    /*Failsafe Behavior*/
    Attempt_Reconnection
    (
      Reconnection_Failed (LT_20_Seconds |
                           GT_20_Seconds Controlled_Crash)
      Reconnection_Successful Return_to_Launch
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target      FROM Swarm,
      $b: Assess_Detected_Object     FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Geofence Breached

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
```

```
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Geofence_Breached
    /*Failsafe Behavior*/
    Take_Manual_Control
    (
      Manual_Control_Failed Controlled_Crash |
      Return_to_Launch
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – GCS Loss of Link

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    GCS_Loss_of_Link
```

```
    /*Failsafe Behavior*/
    Attempt_Reconnect
    (
      Reconnected Resume_Target_Search |
      Not_Reconnected (LT_2_Minutes Wait | GT_2_Minutes Auto_Land )
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Payload Loss of Link

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /*Failure Mode*/
    Payload_Loss_of_Link
    /*Failsafe Behavior*/
    Controller_Reboot
    (
      ( Failure_Remains
        ( GCS_Loss_of_Link Auto_Land |
          Take_Manual_Control Return_to_Launch
        )
      )
```

```
      Failure_Mitigated Resume_Target_Search
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Bingo Fuel

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /* Failure Mode */
    Bingo_Fuel
    /* Failsafe Behavior */
    Check_Fuel_Level
    (
      Is_Returnable Return_to_Launch |
      Is_Not_Returnable Auto_Land
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
```

```
      OD;

COORDINATE <!> $a: Track_Target      FROM Swarm,
      $b: Assess_Detected_Object     FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission - All Failure Modes

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search )

    /*******Failure Modes******/

    /*Control Surface Failure*/
    Control_Surface_Failure Reset_Actuation_Module
    (
      ( Failure_Remains Component_Actuation
        ( Failure_Remains Controlled_Crash |
          Failure_Mitigated Return_to_Launch
        )
      ) | Failure_Mitigated Return_to_Launch
    ) |

    /*Autopilot Failure*/
    Autopilot_Failure Reset_Autopilot_Module
    ( Failure_Remains Attempt_Manual_Control
      ( Manually_Controlled Return_to_Launch |
        Not_Manually_Controlled Controlled_Crash
      ) |
      Autopilot_Restored Return_to_Launch
    ) |

    /*Loss of GPS Failure*/
```

```
    Loss_of_GPS Reset_GPS_Module Attempt_Reconnect
    ( Reconnected Return_to_Launch | Not_Reconnected
      ( LT_20_Seconds Wait | GT_20_Seconds Controlled_Crash )
    ) |

    /*Geofence Breached Failure*/
    Geofence_Breached Attempt_Manual_Control
    ( Manually_Controlled Return_to_Launch |
      Not_Manually_Controlled Controlled_Crash
    ) |

    /*GCS Loss of Link Failure*/
    GCS_Loss_of_Link Reset_GCS Reset_GCS_Module Attempt_Reconnect
    ( Reconnected Resume_Target_Search |
      Not_Reconnected ( LT_2_Minutes Wait |
                        GT_2_Minutes Auto_Land )
    ) |

    /*Payload Loss of Link Failure*/
    Payload_Loss_of_Link Reset_Payload_Module
    ( Failure_Remains Attempt_Manual_Control
      ( Manually_Controlled Return_to_Launch |
        Not_Manually_Controlled Auto_Land
      ) | Failure_Mitigated Resume_Target_Search
    ) |

    /*Bingo Fuel Failure*/
    Bingo_Fuel Check_Fuel_Level
    ( Adequate_Fuel Return_to_Launch |
      Inadequate_Fuel Auto_Land )
    )

*);

Auto_Land:
  Track_Current_Position
  Reduce_Altitude
  Enter_Reverse_Helix
  Land
  Power_Down;

Controlled_Crash:
  Track_Current_Position
  Kill_Throttle;

Return_to_Launch:
  Clearout_Existing_Waypoints
  Set_Final_Waypoint
  Travel_to_Final_Waypoint;

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Bingo Fuel: UAV Sacrifice

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /* Failure Mode */
    Bingo_Fuel
    /* Failsafe Behavior */
    Reassess_Environment
    (
      ( Target_Found Track_Target) |
      ( Non_target_Found Check_Fuel_Level
        (
          Is_Returnable Return_to_Launch |
          Is_Not_Returnable Auto_Land
        )
      )
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm SAR Mission – Bingo Fuel: UAV Relief

```
SCHEMA Swarm_Search_and_Track

ROOT Swarm_Operator:
   Command_Swarm_to_Commence_Mission
   Monitor_Mission
   Issue_Recovery_Command;

Monitor_Mission:
(*
  Assess_Detected_Object
    ( Object_is_Valid_Target | Object_is_Not_Target )
*);

ROOT Swarm:
     {+ UAV +};
     UAV: Commence_Mission
          Search_and_Track_Objects
          End_Mission;

Search_and_Track_Objects:
(*
  (
    Detect_Object Evaluate_Object
    ( Target_Found Track_Target |
      Non_target_Found Resume_Target_Search ) |

    /* Failure Mode */
    Bingo_Fuel
    /* Failsafe Behavior */
    Request_Relief
    Nearest_UAV_Arrives
    Check_Fuel_Level
    (
      Is_Returnable Return_to_Launch |
      Is_Not_Returnable Auto_Land
    )
  )
*);

COORDINATE $a: Command_Swarm_to_Commence_Mission FROM Swarm_Operator
      DO COORDINATE <!> $b: Commence_Mission     FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

COORDINATE <!> $a: Track_Target     FROM Swarm,
      $b: Assess_Detected_Object    FROM Swarm_Operator
            DO ADD $a PRECEDES $b; OD;

COORDINATE $a: Issue_Recovery_Command     FROM Swarm_Operator
      DO COORDINATE <!> $b: End_Mission   FROM Swarm
            DO ADD $a PRECEDES $b; OD;
      OD;

ROOT Environment: (* Present_Object_Signature *);

COORDINATE <!> $a: Present_Object_Signature FROM Environment,
      $b: Detect_Object                     FROM Swarm
      DO ADD $a PRECEDES $b; OD;
```

Swarm vs. Swarm – Event Trace Example

```
SCHEMA Blue_vs_Red
Swarm: {+ UAV +};
UAV: Attack Shoot_enemy_UAV (destroyed | return_to_base);
Shoot_enemy_UAV: (hit | miss);
```

```
ROOT Blue: Swarm;
ROOT Red: Swarm;
COORDINATE <!> $h: hit FROM Blue,
           <!> $d: destroyed   FROM Red
      DO ADD $h PRECEDES $d; OD;

COORDINATE <!> $h: hit FROM Red,
           <!> $d: destroyed     FROM Blue
      DO ADD $h PRECEDES $d; OD;
```

LIST OF REFERENCES

Arquilla, John, and David Ronfeldt. 2013. *Swarming and the Future of Conflict.* Santa Monica, California: Rand Corporation.

Auguston, Mikhail. 2014. *Behavior Models for Software Architecture.* Monterey, CA: Naval Postgraduate School.

———. 2009. *Monterey Phoenix: Modeling Software and Systems Architecture*. Monterey, CA: Naval Postgraduate School.

———. 2016. *System and Software Architecture and Workflow Modeling Language Manual (Version 2).* Monterey, CA: Naval Postgraduate School.

Auguston, Mikhail, and Kristin Giammarco. 2015, Sep. 23. *Monterey Phoenix Home*. https://wiki.nps.edu/display/MP/Monterey+Phoenix+Home.

———. 2016, Jun. 13. *Monterey Phoenix Analyzer App*. http://firebird.nps.edu.

Caswell, Greg, and Ed Dodd. 2014. *Improving UAV Reliability*. DFR Solutions. http://www.dfrsolutions.com/wp-content/uploads/2014/09/Improving-Unmanned-Aerial-Vehicle-UAV-Reliability.pdf.

Dyson, George. 1997. *Darwin among the Machines: The Evolution of Global Intelligence*. Boston, MA: Addison-Wesley Longman Publishing.

Farah-Stapleton, Monica, and Mikhail Auguston. 2013. *Behavior Modeling of Software Intensive System Architectures. Procedia Computer Science* 20: 270–276. http://www.sciencedirect.com/science/article/pii/S1877050913010727.

Frantz, Natalie. 2005. ”Swarm Intelligence for Autonomous UAV Control.” Master’s thesis, Monterey, CA: Naval Postgraduate School.

Freeman, Paul, and Gary Balas. Jun 2014. *Actuation Failure Modes and Effects Analysis for a Small UAV.* America Control Conference.

Garcia, Jorge. 2015. ”Un-building Blocks: A Model of Reverse Engineering and Applicable Heuristics.” PhD dissertation, Monterey, CA: Naval Postgraduate School.

Giammarco, Kristin. 2012. “Architecture Model Based Interoperability Assessment.” PhD dissertation. Monterey, CA: Naval Postgraduate School.

———. “Monterey Phoenix Home” August 14, 2014. Accessed January 1, 2015. https://wiki.nps.edu/display/MP.

Giammarco, Kristin, Mikhail Auguston, Clifton Baldwin, Ji'on Crump, and Monica Farah-Stapleton. 2014. *Controlling Design Complexity with the Monterey Phoenix Approach*. Philadelphia, PA: Complex Adaptive Systems.

Giammarco, Kristin, and Mikhail Auguston. November 13 – 15, 2013. *Well, You Didn't Say Not to! A Formal Systems Engineering Approach to Teaching an Unruly Architecture Good Behavior*. Baltimore, MD: Complex Adaptive Systems Conference.

Giammarco, Kristin, Spencer Hunt, and Clifford Whitcomb. 2015. *An Instructional Design Reference Mission for Search and Rescue Operations.* Monterey, CA: Naval Postgraduate School, Department of Systems Engineering.

Gore, Ross, and Paul Reynolds, Jr. 2008. "Applying Causal Inference to Understand Emergent Behavior." *Proceedings of the 40th conference on Winter Simulation*, 712–721.

Hunt, Spencer. 2015. "Model-based Systems Engineering in the Execution of Search and Rescue Operations." Master's thesis, Monterey, CA: Naval Postgraduate School.

Jackson, Daniel. 2006. *Software Abstractions: Logic, Language, and Analysis.* Cambridge, MA: The MIT Press

Johnson, Chris. 2012. *Technical Challenges For Small UAV Payloads.* Electronic Military and Defense. Accessed May 24, 2016. http://electronicsmilitarydefense.epubxp.com/i/78772–2nd-edition.

Mellodge, Patricia, and Pushkin Kachroo. 2008. *Model Abstraction in Dynamical Systems: Applications to Mobile Robot Control*. Berlin: Springer.

Minar, Nelson, Roger Burkhart, Christopher Langton, Manor Askenazi. 1996. *The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*. Santa Fe Institute, Santa Fe.

Palmer, John, and Kristin Giammarco. 2014. *Utilization of Monterey Phoenix Modeling to Expedite Cost, Estimation in Manufacturing Systems.* Monterey, CA: Naval Postgraduate School.

Pilcher, Joanne. 2015. "Generation of Department of Defense Architecture Framework (DODAF) Models Using the Monterey Phoenix Behavior Modeling Approach." Master's thesis, Monterey, CA: Naval Postgraduate School.

Rabbath, Camille, and Nicolas Léchevin. 2010. *Safety and Reliability in Cooperating Unmanned Aerial Systems*. Hackensack, NJ: World Scientific.

Steward, Victoria. 2015. “Functional Flow and Event-Driven Methods for Predicting System Performance.” Master’s thesis, Monterey, CA: Naval Postgraduate School.

Whitlock, Craig. June 2014. “When Drones Fall from the Sky.” *Washington Post*. http://www.washingtonpost.com/sf/investigative/2014/06/20/when-drones-fall-from-the-sky/

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California